CSER 2011 Spring Meeting - June 21, 2011
Dunning Hall, Room 14, Queen's University

Agenda and Registration Form

The Consortium for Software Engineering Research (CSER) Spring 2011 Meeting will be held on Tuesday, June 21, 2011 and will take place at Dunning Hall, ROOM 14, Queen's University in Kingston.

CSER-Kingston offers a great opportunity to get an update on the latest and greatest software engineering research throughout Canada in a wonderful city.

This year's CSER will feature a set of mini-keynote talks by several new Canadian Software Engineering professors. CSER will also have two parallel interactive workshops, several talks, and a reception with over 20 posters. Please plan to attend and to actively participate!

Would each Researcher/Principal Investigator and industrial participant **please register your group by June 13, 2011 using the form on the next page. At the same time please email Ying Zou <ying.zou@queensu.ca> telling her how many faculty and students from your group will be attending. We need this so we can plan for the food.**

Students: Please remind your professor about the above!

The registration fee for CSER Spring 2011 Meeting remains $150 for faculty members and industrial participants and $50 for students. In the past we had some difficulties in collecting the fees. We would really appreciate an expedient and timely submission of forms and payments.

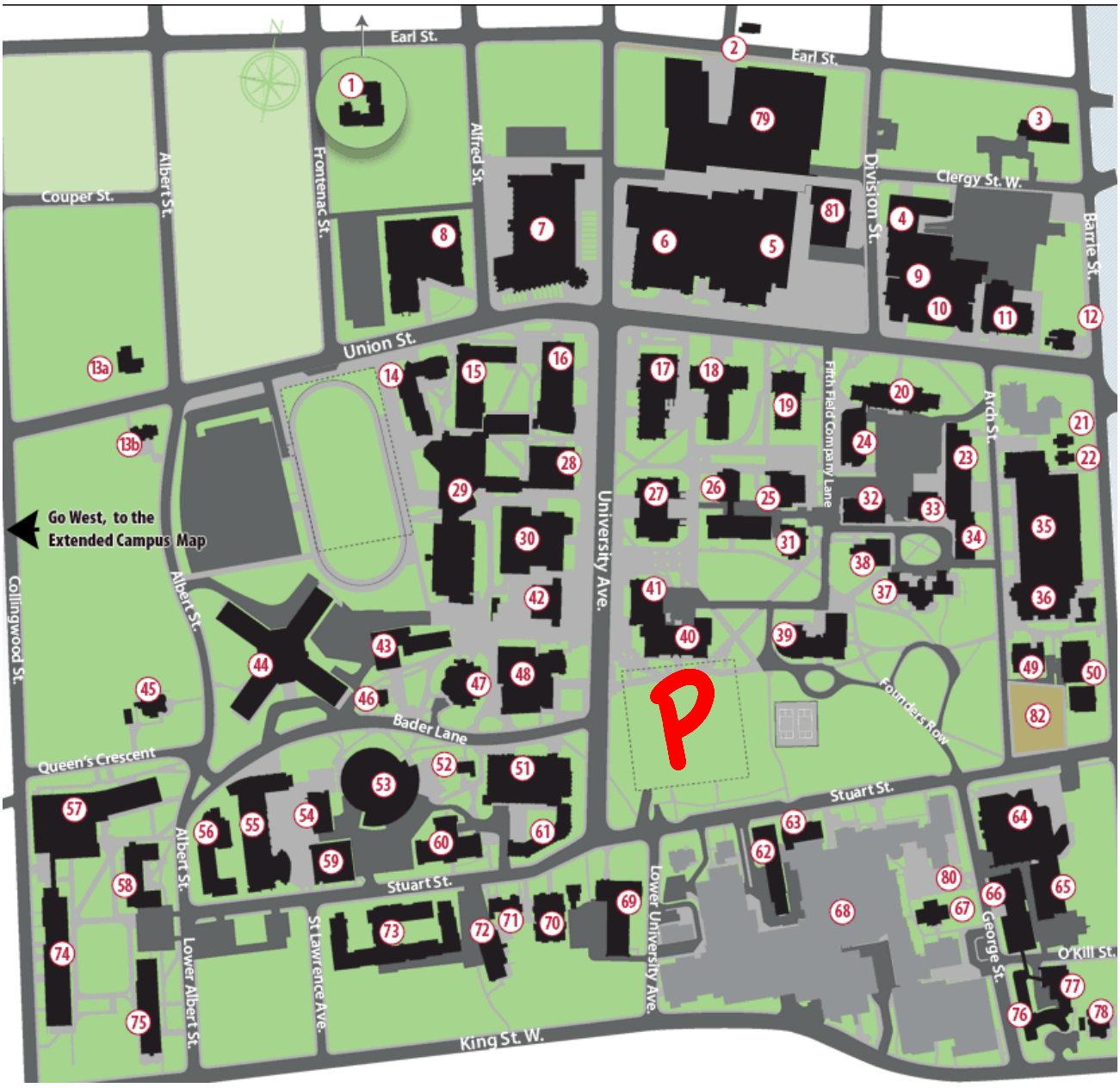
The poster session is organized by Dr. Foutse Khomh < foutse.khomh@queensu.ca>. Contact Foutse if you have any questions regarding posters.

Queen's campus map is shown in page 2 (also available at http://www.queensu.ca/campusmap/). The talks will be taken place in Dunning Hall (i.e., building 16 in the map). The poster session and the reception will be held in Bioscience Complex Atrium (i.e., building 36 in the map). Paid day parking is available and marked as P in the attached map.

We look forward to seeing you on June 21st in Kingston!

Ying (Jenny) Zou
Spring 2011 CSER meeting organizer

Queen's University Campus Map



Accommodation

Queen's University offers economical and premium rooms on campus.

For example, Leggett Hall - Premium air-conditioned 2-bedroom units at the rate of $99.00 for 2-bedroom unit (two rooms with one bathroom). 13% HST is extra.

You can find other rates for single rooms without air condition and the reservation information at http://eventservices.queensu.ca/4/conferencesandgroups.asp.

CSER 2011 Spring Meeting Registration Form
Instructions: By June 13th, Please:

Fax the following completed form to **1-613-482-4708**

Written confirmation of registration will be e-mailed. To facilitate meeting planning activities, it would be appreciated if the registration forms are received no later than **June 13th, 2011**.
Your details (please duplicate if making multiple registrations)

Name: (Family)_____(Given)_____

Organization:_____

Mailing Address:_____

City: _____ State/Province: _____ Zip/Post:_____

Country: _____

Telephone: _____ Fax: _____

Email Address:_____

Names of Students:

1) _____ 2) _____

3) _____ 4) _____

5) _____ 6) _____

7) _____ 8) _____

Calculate fees Note: All Fees are in Canadian Dollars.

	Fee	**Number of Persons**	**TOTAL**
Faculty, industry or Govt.	**$150**		
Students	**$50**		
TOTAL			

Make payment by credit card: __ Visa __ MasterCard

Card Number: _____Expiration Date: Month _____ Year: ____

Cardholder Name: _____

Signature: _____

CSER Spring Meeting 2011 Agenda: Tuesday June 21 2011
Dunning Hall, Room 14, Queen's University

7:30 - 8:30 Board Meeting

8:00 - 8:30 Continental Breakfast

8:30 - 8:45 Welcome and Introduction (Ying Zou, Hausi Müller)

8:45 - 10:30 Keynotes by Young Faculty Members,

Chaired by Hausi Müller, University of Victoria

- **Jeremy Bradbury, University of Ontario Institute of Technology**
  Producing High Quality Concurrent Software
  In general, the development of high quality concurrent code is more difficult than the development of high quality sequential code. One reason for this difficulty is the many different, possibly unexpected, executions of a concurrent program. This talk will overview the current state-of-the-art in software quality assurance of concurrent software. In particular, concurrency testing, model checking and static analysis techniques will be discussed.
- **Lin Tan, University of Waterloo**
  Leveraging Software Semantic Information To Improve Software Reliability
  Software bugs greatly hurt software reliability. In this talk, I will present our recent research that leverages software semantic information in program comments, source code, and commit logs to automatically detect software bugs and understand software developers. We automatically extract specifications from source code and code comments written in a natural language, and use these specifications to detect comment-code inconsistencies, i.e., software bugs and bad comments.

  Detecting software bugs requires a good understanding of developers. Therefore, we study comment semantics and characteristics to understand what developer write in comments, how we can utilize the comments, and what important problems/limitations they reveal which can guide the design of new languages and tools for improving reliability, programmer productivity, software evolution, etc. Additionally, we analyze developers' commit metadata to understand the correlation between a commit's "bugginess" and its social characteristics such as the time of day of the commit, the day of week of the commit, and the experience and commit frequency of the commit authors.
- **Chanchal Roy, University of Saskatchewan**
  Code Clone Detection and Management: Past, Present and the Future
  Reusing code fragments by copying and pasting with or without minor adaptation is a common activity in software development. As a result software systems often contain sections of code that are very similar, called code clones. Previous research shows that a significant fraction (between 7% and 23%) of the code in a typical software system has been cloned. While such cloning is often intentional and can be useful in many ways, it can be also be harmful in software maintenance and evolution. Detection and management of code clones thus becomes an active

and interesting research topic in recent years. In this talk, I will first present the state of the art in clone detection and management including what we have been doing in our group and then will outline the future trends in the area.

- **Reid Holmes, University of Waterloo**
  Improving Comprehension of Source Code Changes
  Software engineering is a human-centric activity. In this talk I will provide an overview of the research I perform with specific emphasis on a new project to that helps developers understand the dynamic impact of the changes they are making to their systems. While developers often have a strong understanding of the static nature of their changes, the dynamic effects of these changes on the runtime behaviour of the program can be harder to comprehend. This approach automatically classifies the impact of a developer's change so they can better understand the dynamic consequences their modification tasks. The overall goal of this project is to enable developers to reason about the dynamic behaviour of their systems in a way that helps prevent unintended behavioural changes from being made.

10:30-11:00 Break

Morning Talks ***Chaired by Kenny Wong, University of Alberta***

11:00-11:20 Juergen Dingel, Queen's University
Component-based development of reactive systems using protocol state machines and model checking
Interfaces represent abstractions which are supposed to facilitate the correct use of an entity by listing the data and operations that the entity makes available and separating its externally visible parts from the internal ones. Arguably, this notion is one of the great success stories in computer science. To further increase the utility of interfaces, numerous proposals have been made to enrich them with more specific information about how the interface elements are to be used. In this talk, I will discuss the potential of protocol state machines (PSMs) for facilitating the model-driven development of component-based systems in general and of reactive systems in particular. I will summarize our recent work on using model checking for determining the compatibility of a component with respect to interface specifications using PSMs.

11:20-11:40 Scott Grant, Queen's University
Visualizations to Support Concept Location
We explore the information provided by concept location techniques like Latent Dirichlet Allocation through three distinct visualizations. Using this information, we demonstrate a relationship between topic models and co-maintenance history. We also explain how these views of source code can give insight about the semantic architecture of the code, and why this information is actually important for software maintenance.

11:40-12:00 Discussion

12:00 - 13:00 Lunch

Afternoon Talks ***Chaired by Jeremy Bradbury, University of Ontario Institute of Technology***

13:00 - 13:20 Kelly Lyons, University of Toronto
Collaborative Decision Making
In this University of Toronto / SAP joint project we are investigating tools and mechanisms to support cross-site collaborative decision making. The goal of this research is to understand the phenomenon of collaboration within decision making, the use of social media to support collaboration in organizations, and the related issue of a changing workforce, and the corresponding impacts on both current and future products and services offerings. In this presentation, we describe the results of three studies: a survey of young people that provides some insight into the changing attitudes, behaviours, and norms of the future workforce and the impact of these changes on decision making; an analysis of collaborative user feedback mechanisms and the impact on software development decisions; and, an implementation and evaluation of a group decision-making technique that has been integrated into SAP StreamWork. We also present next steps and future research

13:20 - 13:40 Hua Xiao, Queen's University
End-users Driven Service Composition for Constructing Personalized Service Oriented Applications
Service composition integrates existing services to fulfill specific tasks using a set of standards and tools. However, current service composition techniques and tools are mainly designed for SOA professionals. It is challenging for end-users without sufficient service composition skills to compose services. In this presentation, we propose a framework that supports end-users to dynamically compose and personalize service recommendation to meet the goals of their daily activities. Instead of requiring end-users to specify detailed steps for the service composition, our framework only requires the end-users to specify the goals of their desired activities using a few keywords to generate an ad-hoc process. To acquire the desired data for service composition, we propose an approach to extract data needed for service composition from existing commercial applications on the Web. In addition, to provide personalized service recommendation, we present an approach to discover desired services for end-users based on the context of end-users. A set of case studies are conducted to evaluate our proposed approaches. The results show that our approaches can effectively extract process knowledge, recommend the desired services for end-users, and generate ad-hoc processes with relatively high precision and recall.

13:40 - 14:00 Hausi Muller and Norha M. Villegas, University of Victoria
Designing smart software systems: Context, control and run-time validation
The continuous evolution from goods-centric to service-centric businesses requires new and innovative approaches for building, running, managing and evolving smart business applications. The complexity of these modern, decentralized, user-centric and distributed computing systems presents significant challenges for businesses. End-users increasingly demand that businesses provide smart software systems that are flexible, resilient, location-based, service-oriented, decentralized, energy-efficient, self-healing. A system with such dynamic properties must be able reason at run-time about its state, environment, and goals. One of the most promising approaches to achieving such properties is to equip software systems with feedback control and effective context management to deal with inherent system dynamics. Applications range from smart web services and location business intelligence to adaptive cloud scheduling and system diagnosis. Our innovative approach to orchestrate run-time system adaptations is based on dynamic context and control theory and involves run-time verification and validation. In the first part of this talk we will present selected

challenges that the software engineering research community must face to enable software systems with intelligent capabilities to address some of these system dynamics. In the second part of the talk, we will present an e-commerce case study where feedback loops and dynamic context management techniques provide effective instrumentation to realize smart interactions as proposed in the vision of the Personal Web.

14:00 - 14:20 Mehdi Amoui, University of Waterloo
Graph-based Runtime Adaptation Framework (GRAF)
One approach for achieving runtime adaptability in software is to use application frameworks that are tailored for the development of self-adaptive systems. In a collaborative research with University of Koblenz-Landau, we designed and developed the Graph-based Runtime Adaptation Framework (GRAF), which enables adaptivity by creating, managing, and interpreting graph-based runtime models of software. GRAF is especially suited for the migration of legacy applications towards adaptive software and attempts to reduce necessary changes to the original software. A scenario of evolving a legacy game engine towards providing adaptive behaviors will be presented to exemplary show how to achieve runtime adaptively with GRAF.

14:20 - 15:00 Discussion

15:00 - 15:30 Break

15:30 - 17:30 Workshop on the Future Trends of Detection, Evolution, Management and Applications of Code Clones, Chaired by Chanchal Roy

15:30 - 17:30 Workshop on Cloud Computing, Chaired by Wendy Powley

17:30 - 17:45 Wrap up

18:45 - 21:30 Poster Session and Reception at Bioscience Complex Atrium (building 36 in the map)

Workshop on Cloud Computing

Chaired by *Wendy Powley, Queen's University*

Talks:

Azada Khalaj, University of Western Ontario
A Proxy-Based Mobile Computing Infrastructure

Proxies can be used as gateways between cloud resources and mobile devices to deal with the challenges resulted from disconnections and the limited resources of mobile devices. This research describes a proxy-based infrastructure that takes into account the proximity between mobile device and proxy and the mobility of the client mobile device. Furthermore, proxies are chosen dynamically and the services provided by a proxy are dynamically changing based on the requirement of the clients for better resource utilization. Several experiments are carried out to evaluate the effectiveness of the proposed infrastructure. The results suggest that the services offered by the proxy can be used for quick recovery after disconnections with only the minimal addition of overhead.

Kamran Sartipi, McMaster University
Identifying Distributed Features in SOA by Mining Dynamic Call Trees

Distributed nature of web service computing imposes new challenges on software maintenance community for localizing different software features and maintaining proper quality of service as the services change over time. In this paper, we propose a new approach for identifying the implementation of web service features in a service oriented architecture (SOA) by mining dynamic call trees that are collected from distributed execution traces. The proposed approach addresses the complexities of SOA-based systems that arise from: features whose locations may change due to changing of input parameters; execution traces that are scattered throughout different service provider platforms; and trace files that contain interleaving of execution traces related to different concurrent service users. In this approach, we execute different groups of feature-specific scenarios and mine the resulting dynamic call trees to spot paths in the code of a service feature, which correspond to a specific user input and system state. This allows us to focus on a the implementation of a specific feature in a distributed SOA-based system for different maintenance tasks such as bug localization, structure evaluation, and performance analysis. We define a set of metrics to assess structural properties of a SOA-based system. The effectiveness and applicability of our approach is demonstrated through a case study consisting of two service-oriented banking systems.

Gaston Keller, University of Western Ontario
Data centre management: VM Relocation Problem

Data centres are complex computer systems composed of thousands of physical servers, which in turn host a number of virtual servers running user applications. The service demand of these applications is dynamic and when it changes, the resource demand of the virtual servers hosting the applications changes accordingly. If a virtual server demands more resources than its host (physical server) can provide, the virtual server has to be relocated (i.e. migrated or replicated) to another host that can satisfy the virtual server’s resource needs. The

focus of our current work is the development of a flexible algorithm that can find a sequence of virtual server relocations with the aim of reducing the resource utilization of overloaded physical servers in a data centre.

Mohammad Hamdaqa, University of Waterloo
Towards a Cloud Application Modeling Language

Standardizing the process of cloud application development is currently a vibrant concern. Although cloud applications share many concepts with existing development paradigms such as real time, service oriented and distributed computing; cloud applications have their own identity, which developers need to understand in order to develop efficient applications on the cloud. The problem with existing software architectures and programming models is that first, they have no holistic view that combines all previous paradigms together, and second they lack many elements that are needed to model and develop cloud applications efficiently. Consequently, there is a need to capture and model elements that can address the on-demand nature of cloud applications within their virtual environment, as well as elements that address scalability, elasticity, high-availability, fault tolerance and billing. After studying a number of cloud platforms, we realized that all these platforms are sharing a common hidden architecture and concepts. Clearly, there is a need to extract this architecture and to find the ontology that relates cloud concepts with each other in order to facilitate the communication between cloud stakeholders. The ontology that will be presented is a part of our research group ongoing work towards defining a platform-independent cloud application modeling language.

Patrick Martin, Andrew Brown, Wendy Powley, Queen's University
Jose Luis Vazquez-Poletti, Universidad Complutense de Madrid
Autonomic Management of Elastic Services in the Cloud

Cloud computing, with its support for elastic resources that are available on an on-demand, pay-as-you-go basis, is an attractive platform for hosting Web-based services that have variable demand, yet consistent performance requirements. Effective service management is mandatory in order for services running in the cloud, which we call elastic services, to be cost-effective. In this paper we describe a management framework to facilitate elasticity of resource consumption by services in the cloud. We extend our framework for services management with the necessary concepts and properties to support elastic services.

Workshop on the Future Trends of Detection, Evolution, Management and Applications of Code Clones

Chaired by: Chanchal K. Roy, University of Saskatchewan, croy@cs.usask.ca

Abstract: Reusing a code fragment by copying and pasting with or without minor modifications is a technique frequently used by programmers, and thus software systems often have duplicate fragments of code in them. Such duplicated fragments are called code clones or simply clones. Although cloning is beneficial in some cases and often programmers intentionally use it, it can be detrimental to software maintenance. For example, if a bug is detected in a code fragment, all the fragments similar to it should be investigated to check for the same bug, and when enhancing or adapting a piece of code, duplicated fragments can multiply the work to be done. A recent study that works on industrial code shows that inconsistent changes to code duplicates are frequent and lead to severe unexpected behaviuor. In this workshop, first, we plan to discuss about the future trends of detecting clone clones, in particular we plan to talk about in efficiently detecting near-miss (Type 3) and semantic (Type 4) code clones. Second, we plan to focus on the insights of clone evolution in order both to find out the effects of clones in software maintenance, and the related issues of how to manage these clones during the evolution. Third, we plan to gather a set of features for a comprehensive clone management system along with the possible technologies of how to build such a clone management system. Finally, we plan to discuss the possible applications of clone detection in other areas of software engineering and vice versa.

Format of the workshop:
The workshop will be purely interactive in nature. Each speaker will talk about four to seven minutes on his/her position on code clones towards the objectives of the workshop and then will lead a discussion for about another three to six minutes on that topic and related issues. Each speaker will come up with one or two important questions related to their talk and these questions will work as the vehicle for the subsequent discussion. Depending on the availability of time, some speakers (especially the seniors ones) might get few extra minutes to lead the discussion.

Invited Speakers cum panelist:
Bram Adams, Queen's University, bram@cs.queensu.ca
Liliane Barbour, Queen's University, 4lb3@queensu.ca
Jeremy S. Bradbury, University of Ontario Institute of Technology (UOIT), Jeremy.Bradbury@uoit.ca
James R. Cordy, Queen's University, cordy@cs.queensu.ca
Massimiliano Di Penta, University of Sannio, dipenta@unisannio.it
Michael Godfrey, University of Waterloo, migod@uwaterloo.ca
Scott Grant, Queen's University, scott@cs.queensu.ca
Daqing Hou, Clarkson University, dhou@clarkson.edu
Andrian Marcus, Wayne State University, amarcus@wayne.edu
Doug Martin, Queen's University, doug@cs.queensu.ca
Philipp Schügerl, Concordia University, philipp.schuegerl@gmail.com
Minhaz F. Zibran, University of Saskatchewan, mfz946@mail.usask.ca

Posters

Poster 1: Scott Grant (Queen's University): ***Visualizations to Support Concept Location***

We explore the information provided by concept location techniques like Latent Dirichlet Allocation through three distinct visualizations. Using this information, we demonstrate a relationship between topic models and co-maintenance history. We also explain how these views of source code can give insight about the semantic architecture of the code, and why this information is actually important for software maintenance.

Poster 2: Martin Mwebesa (University of Ontario Institute of Technology): ***Using static analysis to detect concurrency design patterns***

We propose a static analysis technique to automatically detect concurrent software design patterns in Java source code. First, we identify general characteristics (roles) for a given concurrency design pattern and next we use TXL, a source transformation language, to match the design pattern roles with an actual instance of the design pattern. We evaluate our TXL-based detection technique and assess the recall and precision with respect to each concurrency design pattern. In our evaluation 8 concurrency design patterns are considered: the Single Threaded Execution pattern; the Balking pattern; the Read Write Lock pattern; the Guarded Suspension pattern; the Two Phase Termination pattern; the Lock Object pattern; the Producer consumer pattern and the Scheduler pattern. This evaluation is performed on open source java code examples that contain instances of these 8 patterns. To measure the precision of our technique, we use mutation on the java code examples and revaluate our technique on the mutated code.

Poster 3: Liliane Barbour, Foutse Khomh and Ying Zou (Queen's University): ***Late Propagation in Software Clones***

Two similar code segments, or clones, form a clone pair within a software system. The changes to the clones over time create a clone evolution history. In this work we study late propagation, a specific pattern of clone evolution. In late propagation, one clone in the clone pair is modified, causing the clone pair to become inconsistent. The code segments are then re-synchronized in a later revision. Existing work has established late propagation as a clone evolution pattern, and suggests that the pattern is related to a high number of faults. In this study we examine the characteristics of late propagation in two long-lived software systems using the Simian and CCFinder clone detection tools. We define 8 types of late propagation and compare them to other forms of clone evolution. Our results not only verify that late propagation is more harmful to software systems, but also establish that some specific cases of late propagations are more harmful than others. Specifically, two cases are most risky: (1) when a clone experiences inconsistent changes and then a re-synchronizing change without any modification to the other clone in a clone pair; and (2) when two clones undergo an inconsistent modification followed by a consistent change that modifies both the clones in a clone pair.

Poster 4: Ali Fatolahi (University of Ottawa): ***TOWARD REUSABILITY IN WEB MODELING Using QVT Relations***

In this poster, a model-driven approach for web development is presented. The approach contains two important elements that serve reusability: an abstract model and a set of transformations. Transformations act as the chaining feature of model-driven development (MDD); that is

transformations add to the value of models by transforming them to those of the desired type. As a standard for developing transformations, QVT relations are used in this paper to specify mappings from a high-level model to an abstract model of web-based applications. This model is abstract since it does not rely on any specific web platform but on the common features of web applications. Having this model and its corresponding transformations, model-driven web development for specific platforms becomes faster and more reusable.

Poster 5: Hamzeh Zawawy (University of Waterloo): ***Root Cause Analysis for Distributed Systems using Statistical Techniques***

The complex interactions and the large amount of heterogeneous log data generated in enterprise environments renders problem diagnosis challenging. In this paper, we propose a root cause analysis framework for distributed systems based on annotated requirements goal models. The monitored computer systems are modeled using goal models, which are represented using weighted first-order logic rules. These rules are used to infer the root cause for failures in the monitored systems. Log data emanating from the monitored systems are used as observation supporting or denying these rules. The proposed framework improves over existing approaches by handling uncertainty in observations, using natively generated log data, and by providing ranked diagnoses. The framework is illustrated using a test environment based on commercial off-the-shelf software systems. Our experimental results offer support to the soundness and scalability of our approach.

Poster 6: Karolina Zurowska (Queen's University): ***Symbolic analysis of UML-RT models***

Model Driven Development (MDD) has been introduced to improve quality of software products and manageability of their development. MDD combines the implementation and successive refinement of models until code can be generated from them automatically. MDD has been used in different domains, but has been most successful for the development of reactive systems. Several MDD tools exist including IBM Rational Software Architect Real Time Edition (IBM RSA RTE), IBM Rational Rhapsody and Scade Suite from Esterel Technologies. Although MDD has been used in the industry, more research is needed, e.g., to determine how to best support MDD with suitable model analyses.

One of the more successful analysis methods is symbolic execution. The method, originally introduced for programs in the 70s, is based on traversal of a control flow graph with symbolic values serving as placeholders for input parameters. The result of such traversal is a symbolic execution tree that encompasses all execution paths along with constraints that must be satisfied to follow a particular path. The original idea of symbolic execution was adapted and applied to concurrent /parallel systems and also to state based models.

In the presented work we introduce a technique for symbolic execution of models developed in IBM RSA RTE, which are in the UML-RT modeling language. The technique builds on the intrinsic modularity of UML-RT models and starts with a symbolic execution of a non-composite submodels. The results, symbolic execution trees, are then composed reflecting the structure of the analyzed UML-RT model. A composite symbolic execution tree, which represents all possible executions of the model, is the basis to perform analyses such as reachability, invariants checking or test case generation.

The most important distinguishing characteristic of the presented technique is the reuse of the symbolic execution results. This is possible at the level of modules in UML-RT models (called capsules) and at the level of action code (that is code in UML-RT State Machines). The first type of reuse guarantees that modules that are used in different configurations are analyzed only once. Reuse of symbolic execution results for action code enables analysis of models with different action code languages.

We present the technique using a case study. We show steps of a symbolic execution of a UML-RT model, as well as the results of the performed analysis. We also report on the experiments with several other UML-RT models.

Poster 7: Ernesto Posse (Queen's University): ***lingenoc: a language definition framework***

The construction of software language processing tools such as interpreters, compilers, analyzers, etc. is labour intensive. Parser generators help relieve the effort required to build tools that deal with concrete syntax. To represent abstract syntax, designers and implementers typically use well-known patterns, in particular, the composite pattern [gamma-et-al:94:design-patterns]. To implement operations on abstract syntax, including interpreters, analysis, pretty-printers, etc. some patterns such as the visitor pattern [gamma-et-al:94:design-patterns] are often used. In this paper we introduce a language definition framework which aims to close the gap between these implementation patterns and the formal specification of abstract syntax and language operations. This language description is given in terms of basic concepts from universal algebra. From this description, code is generated implementing composite and visitor classes for the abstract syntax of a language. A key feature of our framework is the ability to merge and combine languages (both syntax and operations) to define new languages. We have implemented this framework as a Python library, but the central concepts are applicable to other languages.

Poster 8: Eric James Rapos (Queen's University): ***Incremental Test Case Generation for UML-RT Models***

Model driven development (MDD) is on the rise in software engineering and no more so than in the realm of real-time systems. Being able to leverage the code generation and validation techniques made available through MDD is worth exploring. Currently, the existing process of regenerating test cases for a modified model of a system can be costly, inefficient, and sometimes even redundant. It is our goal to work on incrementally generating test cases based on an existing test suite and some change performed on a given model. Currently, test cases can be generated through the use of their symbolic execution trees (SETs). Instead of regenerating test cases directly from the SET, we will compare the SET of a model after some refinement, and determine how this changes the generated test cases for the given model. It is our hope to develop a prototype that will incrementally generate test cases, as opposed to regenerating a test suite every time a model is changed. At the beginning, we will use a catalog of model refinements recently presented in the literature, to hopefully improve the efficiency of test case generation in the presence of model changes. As an end result, it is our goal to present an improved understanding of the impact of typical state machine evolution steps on test cases, and how this impact can be mitigated by reusing previously generated test cases. We are also aiming to implement this in a software prototype to automate and evaluate this process.

Poster 9: Kevin Jalbert (University of Ontario Institute of Technology): ***Predicting how difficult bugs are to detect using source code metrics***

The proposed technique uses static source code metrics to predict the difficulty of detecting bugs within a source code unit. We use mutation testing with mature test suites to estimate how difficult it is to find a bug within the observed source code unit. The estimated difficulty of detecting a bug can be used to train a support vector machine with a feature set of source code metrics. After sufficient training it becomes possible to predict how difficult bugs are to detect using only source code metrics of the observed source code unit. Preliminary results in classifying the difficulty of detecting a bug using only a single open source project had a cross validation accuracy of 76.80%.

Poster 10: Turki Alharkan (Queen's University): ***Intrusion Detection System as a Service (IDSaaS)***

Intrusion Detection System (IDS) is a security technology, which can detect, prevent and possibly react to computer attacks. IDSs have been proven to be effective tools in the conventional local and wide area networks. In a typical network scenario, IDS will generate alerts regarding any security threats and log them for further analysis. Then, a network administrator can decide to rely on the IDS judgment and take an action or let the system react through a predefined plan.

In the Cloud Computing environment, the need for IDS is still essential and irreplaceable. Cloud consumers can not only depend on the cloud's provider security infrastructure, but they need to monitor and protect their virtual existence by implementing IDS along with other security technologies like firewalls, access controls and data encryption within the cloud fabric.

This is an approach to investigate the possibility to provide IDS security method through the concept of Cloud Computing. A key point in deploying IDS in the cloud is to detect all security violations resulted from cyber attacks, whether it is originating from inside or outside the cloud, in an efficient and cost effective manner.

Poster 11: Nicolas Chausse (Queen's University): ***Building a general-purpose language analyzer using a generative language definition framework***

Given a formal specification of a language's abstract syntax and semantic operations, the language definition framework Lingenoc can be used to automatically generate supporting processing tools for that language including pretty printers, compilers, and interpreters. We are interested in the efficient development of analyzers for languages defined with Lingenoc. More precisely, we will build an analyzer that will check programs for different properties related to the reachability of desirable and undesirable states (e.g., deadlock or livelock). The analyzer will be general-purpose in the sense that it will be applicable to all languages defined with Lingenoc without change. We will build a prototype analyzer which will be evaluated on a language used to represent and analyze UML-RT models.

Poster 12: Eyrak Paen (Queen's University): ***Using transformation evolution to compare different model transformation languages***

Transformations play a central role in Model Driven Development. Similar to the development of other types of software, a transformation's specification and implementation does not necessarily remain static over the course of a project's lifetime. The transformation may evolve. We consider in a case study the development of a transformation that converts UML-RT models to a process algebra modelling language called kiltera. The specification of the transformation is defined incrementally. We implement the increments of this transformation in different model transformation languages and compare the artifacts. Our goals are to devise new metrics for comparing model transformation languages and to gain more insight into the design of model transformation languages.

Poster 13: Forough Norouzi (Western Ontario): ***Energy efficient job forwarding***

The number of high density servers in today's data centers have been rapidly growing and consequently, placing greater demands on power consumption of data centers. Greater energy usage results in higher power costs increases cooling costs and maintenance costs of the data center. Thermal aware management of data centers can help address these problems. Focusing in thermal issues causes delay in response time. In this work, a typical data center and its thermal behavior has been simulated and several thermal based job forwarding algorithms have been simulated. These

algorithms has been compared in terms of energy consumption and total delay in job completion. Simulation has been done on real batch type workload. Thermal zone job forwarding as our future work will be introduced.

Poster 14: Foutse Khomh and Ying Zou (Queen's University): ***Predicting Post-release Defects Using Pre-release Field Testing Results***

Field testing is commonly used to detect faults after the in-house (e.g., alpha) testing of an application is completed. In the field testing, the application is instrumented and used under normal conditions. The occurrences of failures are reported. Developers can analyze and fix the reported failures before the application is released to the market. In the current practice, the Mean Time Between Failures (MTBF) and the Average usage Time (AVT) are metrics that are frequently used to gauge the reliability of the application. However, MTBF and AVT cannot capture the whole pattern of failure occurrences in the field testing of an application. In this talk, we present three metrics that capture three additional patterns of failure occurrences: the average length of usage time before the occurrence of the first failure, the spread of failures to the majority of users, and the daily rates of failures. We also present the results of a case study showing that the three metrics complement the traditional MTBF and AVT metrics and can predict the number of post-release defects in a shorter time frame than MTBF and AVT.

Poster 15: John Khalil and Ramiro Liscano (University of Ontario Institute of Technology): ***Software Modelling For Wireless Sensor Network***

Network delay and power constraint requirements are two significant challenges for Wireless Sensor Networks (WSNs). This is more evident in the WSNs where the real-time properties of the network are a significant component of the performance of the WSN. Being able to include and analyze the network delays and power consumption of WSNs at the modeling layer can reduce the cost associated with dealing with these issues at the coding layer. Model analysis detects the design performance behaviour in the early stages of designing and gives the opportunity to enhance the design before the actual code is implemented. Using Marte and SysML allows for the software modeling at higher abstraction layers other than at the coding layer. In this poster, we present a model for WSNs using SysML and Marte, which are profiles of the UML language. Moreover, we also present an example of the use of SystemC and MAST analysis tools to help analyse for network delays and power consumption in a WSN.

Poster 16: Kamran Sartipi (McMaster University): ***Identifying Distributed Features in SOA by Mining Dynamic Call Trees***

Distributed nature of web service computing imposes new challenges on software maintenance community for localizing different software features and maintaining proper quality of service as the services change over time. In this paper, we propose a new approach for identifying the implementation of web service features in a service oriented architecture (SOA) by mining dynamic call trees that are collected from distributed execution traces. The proposed approach addresses the complexities of SOA-based systems that arise from: features whose locations may change due to changing of input parameters; execution traces that are scattered throughout different service provider platforms; and trace files that contain interleaving of execution traces related to different concurrent service users. In this approach, we execute different groups of feature-specific scenarios and mine the resulting dynamic call trees to spot paths in the code of a service feature, which correspond to a specific user input and system state. This allows us to focus on a the implementation of a specific feature in a distributed SOA-based system for different maintenance tasks such as bug

localization, structure evaluation, and performance analysis. We define a set of metrics to assess structural properties of a SOA-based system. The effectiveness and applicability of our approach is demonstrated through a case study consisting of two service-oriented banking systems.

Poster 17: M. Mondal, C. K. Roy, R. K Saha, J. Krinke, and K. A. Schneider (University of Saskatchewan and University College London): ***Comparative Stabilities of Cloned and Non-cloned Code: An Empirical Study***

Code Cloning has been presented as a controversial term in the realm of software engineering research because of its dual but contradictory roles in software maintenance. Many in-depth empirical studies conducted over the past decade and a few existing impact assessment metrics have revealed this dualism. Different researchers have investigated the practical behavior of clones through different keenly analytic approaches. Unfortunately, those investigations ended up with contradictory concluding remarks. In this poster, we present a comprehensive empirical study that aims to analyze the comparative stability of cloned and non-cloned code using three methods associated with a respective set of stability measurement metrics. For detecting clones we have used the recently introduced hybrid clone detection tool NiCad which exhibits both high precision and recall in detecting Type-1, Type-2 and Type-3 clones. Our five dimensional investigation on 12 diverse subject systems written in three different languages considering three different clone types discovers that the candidate methods have disagreements in making decisions regarding the stability of cloned code vs. non-cloned code and that code stability depends on the development strategy of the subject systems to a great extent.

Poster 18: M. F. Zibran and C.K. Roy (University of Saskatchewan): ***Code Clones: Etiology, Effects, and Treatment***

Duplicate or similar fragments in the source code are known as code clones. In this poster, we first address the causes of cloning in software and distinguish the beneficial and detrimental effects code clones. We also study the existence and evolution of clones in 1,636 releases of several real software systems in search of factors (e.g., language, paradigm) affecting the cloning phenomenon. Finally, we introduce an efficient technique for clone detection in IDE, and a constraint programming approach for conflict aware optimal scheduling of code clone refactoring.

Poster 19: Mark Syer (Queen's University): An Industrial Case Study on Supporting the Comprehension of System Behaviour Under Load

Large-scale software systems achieve concurrency on enormous scales using a number of different design patterns. Many of these design patterns are based on pools of pre-existing and reusable threads that facilitate incoming service requests. Thread pools limit thread lifecycle overhead (thread creation and destruction) and resource thrashing (thread proliferation). Despite their potential for scalability, thread pools are hard to configure and test because of concurrency risks like synchronization errors and dead lock, and thread pool-specific risks like resource thrashing and thread leakage. Addressing these challenges requires a thorough understanding of the behaviour of the threads in the thread pool. We argue for a methodology to automatically identify and rank deviations in the behaviour of threads based on resource usage.

Poster 20: Stephen Thomas (Queen's University): ***Mining Software Repositories Using Topic Models***

Software repositories, such as source code, email archives, and bug databases, contain unstructured and unlabeled text that is difficult to analyze with traditional techniques. We propose the use of statistical topic models to automatically discover structure in these textual repositories. This discovered structure has the potential to be used in software engineering tasks, such as bug prediction and traceability link recovery. Our research goal is to address the challenges of applying topic models to software repositories.

Poster 21: Haroon Malik (Queen's University): ***A Methodology to Support Load Test Analysis***

Performance analysts rely heavily on load testing to measure the performance of their applications under a given load. During the load test, analyst strictly monitor and record thousands of performance counters to measure the run time system properties such as CPU utilization, Disk I/O, memory consumption, network traffic etc. The most frustrating problem faced by analysts is the time spent and complexity involved in analysing these huge counter logs and finding relevant information distributed across thousands of counters. We present our methodology to help analysts by automatically identifying important performance counters for load test and comparing them across tests to find performance gain/loss. Further, our methodology help analysts to understand the root cause of a load test failure by finding previously solved problems in test repositories. A case study on load test data of a large enterprise application shows that our methodology can effectively guide performance analysts to identify and compare top performance counters across tests in limited time thereby archiving 88% counter data reduction.

Poster 22: Ripon K. Saha, Chanchal K. Roy, Kevin A. Schneider (University of Saskatchewan): ***The gCad Near-Miss Clone Genealogy Extractor and Classifier***

Extracting code clone genealogies across multiple versions of a program and classifying them according to their change patterns underlies the study of code clone evolution. While there are a few studies in the area, the approaches do not handle near-miss clones well and the associated tools are often computationally expensive. To address these limitations, we present a framework for automatically extracting both exact and near-miss clone genealogies across multiple versions of a program and for identifying their change patterns using a few key similarity factors. We have developed a prototype clone genealogy extractor, applied it to three open source projects including the Linux Kernel, and evaluated its accuracy in terms of precision and recall. Our experience shows that the prototype is scalable, adaptable to different clone detection tools, and can automatically identify evolution patterns of both exact and near-miss clones by constructing their genealogies.