Research Statement

Over 90% of the data available in the world has been generated in the last two years [1]. ‘Big Data’ analytics has become the next hot topic for most companies - from Financial institutions to Technology companies to Service providers. Over the last 9 years I have been working on such Big Data problems:

• **Scientific Data**: During my tenure at the Scientific Data Management (SDM) center [4], I worked on creating and managing the provenance of terabytes of data generated by physicists, climate scientists, and underground water researchers.

• **Operational Data**: My dissertation research was related to analyzing hundreds of gigabytes of log files that were generated by the Virtual Computing Lab, a cloud computing infrastructure that served over 30,000 users, in order to determine the run time operational profile. More recently, during my tenure as a Post Doc at Queen’s University, I have been working with industrial researchers at BlackBerry on analyzing high velocity data generated by large software projects at run time, for performance analysis.

• **Ultra Large Repositories of Software**: My current work is in collaboration with industrial researchers at Avaya Labs (formerly Bell Labs) and Microsoft Research on analyzing ultra large repositories with millions of Open Source Software (OSS) projects. I also collaborate with academic researchers in Europe (ITU Copenhagen and Leipzig University) and North America (Queen’s University and University of Southern California) on analyzing the rating system and revenue models of Android Apps, by mining several versions of over 100,000 apps in the Google Play app store.

More generally, I have focussed on analyzing Big Data from Software Systems for solving Software Engineering (SE) problems. In the remainder of this statement, I present my motivation for studying Big Data, what I have found so far, and what my vision for this area is.

Motivation

Software Engineering (SE) research over the past few decades has identified and attempted to solve a variety of issues pertaining to the development of software. Increasingly, empirical studies on a collection of software projects (between 5 and 10), have been able to shed light on current development, and maintenance practices of software engineers. However, these studies often face the threat of ‘Generalizability’, since only a limited set of projects have been examined. Therefore, the conclusions of these studies may only be applicable to a limited set of software projects. For example, a conclusion about more complex code having more defects in the research study by Zimmermann et al. [14], which was carried out on the dataset from the Eclipse IDE, may only be applicable to large Java-based IDEs. One solution to this problem is to replicate studies to ensure that conclusions are consistent, as in the case of the example above, where the conclusions were confirmed by several other studies. Another solution is to provide the context in which the conclusions are applicable, again as in the example we presented above, where the context of large Java-based IDEs was made explicit in the title ('Predicting Defects for Eclipse'). For either solution, the SE community needs an approach to quantify and place in context the generalizability of case study conclusions.

Hence, along with my collaborators at Microsoft Research [10] (which was published at the 9th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2013), one of the SE research communities top venues), we looked at ways to measure the diversity and representativeness of the case studies in a research project. We proposed a metric called ‘Coverage’ to quantify generalizability. This metric can also be used to place current studies in context, thereby enabling software practitioners to choose the results that are most appropriate to their software project. In this research, we demonstrate our approach by measuring the coverage of the research papers from the past two years of ICSE and FSE (the two top SE conferences) against a universe of open source software projects collected from Ohloh, an online repository of meta-data on such projects. We find that although the coverage is reasonable in certain papers, there is a lot of room for improvement.

Therefore even though studies on a small set of cases are important in any research area, complementary studies that look at entire ecosystems of software projects are needed as well. Hence, the core of my
research is related to mining Big Data with respect to SE (Ultra large repositories of software projects), and analyzing different combinations of software repositories for different stakeholders. By leveraging data from an ultra large number of software projects, we are able to (a) make conclusions that are far more generalizable, (b) identify relationships between software projects in an ecosystem, and (c) identify recurring patterns amongst software projects.

Current Research

Hence, for the past 9 years I have worked on using Big Data in SE to address the concerns of different stakeholders of software systems such as software maintainers and software operators. Some of the projects that I have recently finished are as follows:

- **Developers**: I have done empirical software studies both in the macro level and micro level to provide actionable evidence-based intelligence to developers. At the macro level, we analyzed over 10K C projects in GitHub to determine that there are certain use cases of the goto statement that are not as harmful as Edsger W. Dijkstra stated [8]. We found that developers routinely use the goto statement for error handling and memory cleanup operations since the C language does not have exception handling or finalize methods. Thus we recommend that developers can use goto statements if they are careful and use it for simple purposes. At the micro level, we looked at 21 free-to-download mobile apps from the Google Play store and found that the advertisements in the apps had a sizeable amount of hidden costs [6]. We found that response time increases, battery power decreases, network usage increases, and maintenance costs increase. Hence developers need to be careful about how they use ads in their apps, or they risk losing customers.

- **Software Maintainers**: One recent approach for bug localization is to use Information Retrieval (IR) techniques to calculate the textual similarity between source files and bug reports. However, these IR techniques have many configurable parameters. Currently, most of the research studies select a particular configuration without examining the influence it has on the results. In our study [13] (which was accepted for publication in the IEEE Transactions on Software Engineering, the SE community’s flagship journal), we examine thousands of configurations in order to detect the best set of parameters for IR based bug localization. We also use an ensemble of configurations to improve the accuracy of bug localization.

- **Software Managers**: In our study, we looked at improving the performance of the linear models built on SE data by using local models (built on a subset of the data) instead of global models (built on the whole dataset) [5]. We found that even though local models perform better than global models, the amount of detail available in the local models can overwhelm a manager. Hence we proposed that the managers use a hybrid approach - a global model with local considerations. This study won the ‘Best Paper’ award at the 9th Working Conference on Mining Software Repositories (MSR 2012), and was invited for a journal extension at the Empirical Software Engineering journal.

- **Performance Engineers**: We examined the issues relating to performance problems in multi user systems. Often when there is a performance issue that is found by the performance engineers (specialized software operators) in the load test of software projects, they have to manually associate millions of entries from thousands of resource counters (like CPU and Memory usage) to millions of execution log lines in order to debug. In our study [12] we proposed an automatic approach to link the two data sources (resource counters and execution logs), so that the performance engine, in order to debug, only needs to look at a small subset of the log lines when a performance issue arises. We evaluated our approach on an ultra large scale industrial software project at BlackBerry, and an open source benchmark project.

- **Software Operators**: The execution logs from large scale systems contain hundreds of gigabytes of run time information. This information can be used to identify the operational profile of a system, and the possible root causes of functional and performance issues among other things. In one study [9], we used the Suffix Array data-structure to identify the frequency of patterns (of varying lengths) in linear time for the purpose of operational profiling. Thus we could scale our approach easily to the gigabytes of logs (with millions of log lines) that were generated by a Cloud Computing Infrastructure called Virtual Computing Lab that served the IT needs of over 30,000 users.
Research Vision

Data has always been used in every company irrespective of its domain to improve the operational efficiency and the products themselves. However, analyzing and extracting information from Big Data is the next revolution in technology \[2\], since previously unknown nuggets of information are now made visible. Likewise in software engineering, data collected about the development of software, the operation of the software in the field, and the users feedback on a software have been used before. However, collecting and analyzing this information across hundreds of thousands or millions of software projects gives us the unique ability to reason about the ecosystem at large, and software in general.

At no time in history has there been easier access to extremely powerful computational resources as it is today, thanks to the advances in cloud computing, both from the technology and business perspectives. Therefore, it is easier today than ever before to analyze big data. As part of my future research, I would like to look at Big Data in Software Engineering from two complementary dimensions.

1. Research Opportunities: Over the last few years, both mobile apps and open source distributed development have become hugely popular in the software universe. There are over 700,000 Android apps, and over 900,000 iOS apps to date, with billions of dollars at stake. By analyzing the entire ecosystem of apps, we can identify patterns across these apps. For example, we (in collaboration with Queen’s University, École Polytechnique de Montréal, ITU Copenhagen and Leipzig University, and University of Southern California), looked at over 100,000 apps in the Android market and were able to identify patterns in the ratings of these apps, as well as how the developers use advertisement libraries in these apps [11]. I would like to continue down this line and look at what makes an app successful, how to make the app more visible in the app market, and how to maximize revenue.

On the open source distributed development front, services such as Github have greatly aided in accelerating the number of software projects. As of today, Github alone boasts of hosting over 6.9 million repositories. Any research on such large repositories could potentially impact millions of developers. By analyzing these millions of repositories, we are able to understand the behaviour among developers who collaborate on OSS projects from distributed physical locations. Some of these results may even translate to outsourced corporate software projects. Currently, in collaboration with Avaya Research labs (formerly Bell Labs), we are looking into how the choice of a software build system technology impacts the build maintenance effort by analyzing 800,000 open source repositories in Github \[7\].

2. Research Challenges: Even though there are plenty of opportunities for pursuing research on Big Data in Software Engineering, there are a few challenges that needs to be addressed. One such challenge is the proper application of Data Analysis and Statistical Techniques, especially due to the scale and nature of data (often SE data can be highly skewed). To this end, I am working with Audris Mockus, a world-renowned researcher on empirical software engineering from Avaya Research Labs, to identify and document the best statistical practises for software engineering researchers. We gave a tutorial about this topic at the 9th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2013) titled Statistics in Software Engineering: Pitfalls and Good Practices \[3\]. This tutorial was attended by over 25 people. Several educators requested the slides and other material for their graduate level SE courses.

Another challenge is that of filtering noise from datasets. For example, Github does not distinguish between student projects and user-targeted software. If one aims to study developers, such student repositories need to be filtered out. Another challenge is creating mathematical and statistical solutions for analyzing large datasets that scale linearly. Both time and space constraints are very critical when the analysis is in a large scale. Such solutions for Big Data Challenges in SE have rich research potential, as they can be transferred to Big Data analysis in other research areas as well (and vice versa).

Overall, the field of software engineering research is evolving very quickly. This is in part due to the ubiquitous nature of software. Thus, we as software engineering researchers are at the cusp of redefining “Software Engineering Research”. Even though the field of computer science has been collaborating with medical experts (Biomedical Computing), basic science researchers (Computational Physics, Chemistry, and Math), and financial experts (Trend analysis and forecasting), software engineering research has had limited overlap. Each of the applications in these fields are highly critical, and their quality is paramount. The lessons learnt from decades of software engineering can therefore have a great impact. However, we first need to evaluate such theories on a large sample of software - Big Data in Software Engineering. Therefore I would like to work on these highly inter-disciplinary problems along with researchers, both from other areas of computer science, and from other fields of science and engineering, to leverage the software engineering dimension for improving the applications from the perspective of different stakeholders.
References


