Background and Motivations: Logging is one of the key techniques applied in large-scale software systems. Software developers insert logging code to record system runtime information. This information is then widely used for a variety of purposes, e.g., system monitoring, debugging, remote issue resolution, test analysis, security and legal compliances, and business decision making [1]. According to Gartner, tools for managing log messages are estimated to be a 1.5 billion market and have been growing more than 10 % every year [2]. However, it is very challenging to conduct good quality logging code in a systematic way since there is no established logging guidelines [3]. Logging is a cross-cutting concern, since developers often intermix logging code with feature code. Developers write logging code solely based on their domain expertise and revise them in an ad-hoc fashion [4]. As the pace of software development increases these days, it is more difficult to keep logging up-to-date and sufficient. Out-of-date and insufficient logging could hinder developers' understanding of the system, thus costing time and human labor to diagnose and pinpoint system problems. Therefore, research on improving logging is urgently needed. This field of research has been very active in recent years, as research papers on this topic are published at top tier software engineering conferences every year. However, current research has only touched the tip of the iceberg. We have not come up with a universal logging guideline to be applied in practice, nor have we proposed a robust, usable framework for developers to evaluate and improve their logging code.

Hypothesis: Logging greatly impacts software quality. Historical logging code changes and source code context can be utilized to assist high quality logging code development.

Objective: This research is aimed at providing guidelines and suggestions for good quality logging. Developers can leverage our research to evaluate their current logging practice. They can also use our research tools to automatically compose and revise logging code.

Experimental approach: (1) Empirical study. We intend to evaluate the current logging practice conducted by practitioners. We will analyze open-source software and characterize logging changes from three dimensions. **Where-to-log** focuses on deciding the appropriate logging points. **What-to-log** focuses on selecting the information (variables, texts, etc.) recorded in logging code. **How-to-log** focuses on composing maintainable, readable, up-to-date logging code. This is important as nowadays software evolves rapidly and stale logging code is problematic for many tasks. (2) Evaluation framework. We will design and develop an evaluation framework for practitioners to evaluate the existing logging code. The evaluation criteria will be formalized from the three dimensions mentioned above. We will implement the best logging practices as rules so that whenever there is problematic logging code, we could show warnings and let developers decide whether they should revise it. (3) Suggestion framework. Combining the rich knowledge we extracted from historical logging code changes, we aim to provide logging suggestions to developers. Techniques from data mining and machine learning fields could be used. A potential use-case could be: when a developer is writing code, logging code suggestions will pop out automatically.

Anticipated Significance: The logging evaluation and suggestion framework will revolutionize the ways developers do logging. It will guide developers to decide what-to-log, where-to-log, and how-to-log. In my master's thesis, I have only studied the characteristics of historical changes of logging code and manually found a few anti-patterns. In my future Ph.D study, I plan to extend it in a more systematic way. A thorough understanding of logging code change rationale will be studied and an automated approach to logging code improvement will be proposed.

References: [1]B. Chen and Z. M. Jiang, "Characterizing logging practices in Java- based open source software projects – a replication study in Apache Software Foundation," ESE 2017. [2] Gartner (2014) SIEM Magic Quadrant Leadership Report.. [3] Q. Fu, J. Zhu, W. Hu, J.-G. Lou, R. Ding, Q. Lin, D. Zhang, and T. Xie, "Where Do Developers Log? An Empirical Study on Logging Practices in Industry," ICSE Companion 2014 [4]D. Yuan, S. Park, and Y. Zhou, "Characterizing Logging Practices in Open-source Software," ICSE 2012.