# Effective Mining of Software Repositories

Marco D'Ambros
REVEAL @ Faculty of Informatics
University of Lugano, Switzerland

Romain Robbes
PLEIAD @ Computer Science Department (DCC)
University of Chile, Chile

## I. Summary

With the advent of open-source, the Internet, and the consequent widespread adoption of distributed development tools, such as software configuration management and issue tracking systems, a vast amount of valuable information concerning software development and evolution has become available.

Mining Software Repositories (MSR)—a very active and interest-growing research field—deals with retrieving and analyzing this data. Empirical analyses of software repositories allow researchers to validate assumptions previously based only on intuitions, as well as finding novel theories. In turn, these theories about the software development phenomenon have been translated into concrete approaches and tools that support software developers and managers in their daily tasks.

In this tutorial, we provide an overview of the state of the art of MSR. In particular, we describe what software repositories are, what in turn Mining Software Repositories is, what techniques are available to researchers and practitioners, and finally, what the limitations of MSR are nowadays, and how to fix them.

### A. MSR Approaches

Based on the information available in software repositories, a variety of studies have been performed, and techniques have been proposed to assist the stakeholders of the development process. We present a selection of MSR approaches, grouped in categories. In each case, we explain not only the results of the approach, but also *how* the evaluation was performed, in great detail. The types of approaches we present are:

**Empirical Studies.** We demonstrate how one can use MSR data to perform empirical studies by way of examples; we then extract guidelines on performing empirical studies with software repositories.

**Change Prediction.** Change prediction tackles the problem of identifying entities in a software system that are likely to change next. Software repositories act as both a data source and an evaluation device for change prediction approaches.

**Defect Prediction.** In a world were resources are limited, managers have to optimize the allocation of QA activities, as they cannot afford to give equal attention to each and every source code file. Defect prediction classifies files as potentially defective by considering various attributes of the source code, such as its complexity or its tendency to change.

**Expertise and Bug Assignment.** The information in defect repositories can be used to model the knowledge of developers about the systems they work on. This can be used to recommend a specific expert when help is needed over a particular piece of code; a related problem is automatic bug assignment.

**Code Search.** With the massive amount of code freely accessible on the Internet, there is a fair probability that a given problem has been solved already, and that an implementation is available somewhere. All that is needed, is to find it.

**Visual Evolution Analysis.** One way to understand the large amount of data present in the history of a software system is to visualize it. We present a selection of software visualization tools and approaches that handle repository data, and some of the case studies they were evaluated on.

**Human Aspects.** Software is built by humans, and for humans. As such, taking the human element out of the loop is not advisable. There is some research interest in exploiting developer communication artifacts, such as emails and other free-form text communications (chats, wikis, blogs).

### B. Limitations of MSR

The amount of data available for MSR studies is a boon for empirical research. However, this data comes with strings attached. In this last part, we document the common threats to the validity of empirical studies based on software repositories data. We also go further, and highlight the shortcomings of the current crop of software repositories for empirical research, and discuss some of the possible solutions.

## II. Presenters

**Marco D'Ambros** earned his Ph.D. in October 2010 and is currently a postdoctoral researcher at the University of Lugano, Switzerland. He previously received MSc degrees from both Politecnico di Milano (Italy) and the University of Illinois at Chicago. His research interests lie in the domain of software engineering with a special focus on MSR, software evolution, and software visualization. He authored more than 30 technical papers, and is the creator of several software visualization and program comprehension tools.

**Romain Robbes** is an Assistant Professor at the University of Chile. He earned his Ph.D. in December 2008, from the University of Lugano, Switzerland and received his Master's degree from the University of Caen, France. His research interests lie in Empirical Software Engineering and MSR. He authored more than 30 papers on these topics, including top software engineering venues (ICSE, ASE), and best paper awards at WCRE 2009 and MSR 2011. He is program co-chair of IWPSE-EVOL 2011, and the recipient of a Microsoft SEIF award 2011.