

Learning from the Past

Gail C. Murphy
University of British Columbia, Canada
murphy@cs.ubc.ca

Abstract

Software development activities generate a huge number of artifacts: source code, design documents, requirements specifications, email messages, and so on. Many of these artifacts have been revised many times, and most software development teams retain all of the versions of all of the artifacts generated. The stored revisions of all of these artifacts contain a wealth of historical information. Sadly, software developers do not typically consider this historical information when working on the system because pertinent information is difficult to find in the morass of artifacts available, and even when an artifact of interest is found, it can be difficult to assess if the information in the artifact is sufficiently up-to-date. In this talk, I will describe two projects we are working on that demonstrate developers can cost-effectively learn from the past of their project. The Hipikat project is developing techniques for collating software development artifacts, and for recommending pertinent artifacts based on the context in which a developer is working. The Concern Graph project is developing an approach and tool to help a developer find and capture scattered, crosscutting concerns in source code that are related to a modification task. The tool supports the later detection and repair of inconsistencies between the captured concern description and a subsequent version of the code base. Used together, these two approaches can add value to program understanding and reverse engineering tools and tasks because the time and effort expended by a developer today can be amortized over subsequent uses of that information by other developers in the future. This work is joint with Davor Cubranic and Martin Robillard.