3

Analysing Software Repositories to Understand Software Evolution

Marco D'Ambros[1], Harald C. Gall[2], Michele Lanza[1], and Martin Pinzger[2]

[1] Faculty of Informatics, University of Lugano, Switzerland
[2] Department of Informatics, University of Zurich, Switzerland

Summary. Software repositories such as versioning systems, defect tracking systems, and archived communication between project personnel are used to help manage the progress of software projects. Software practitioners and researchers increasingly recognize the potential benefit of mining this information to support the maintenance of software systems, improve software design or reuse, and empirically validate novel ideas and techniques. Research is now proceeding to uncover ways in which mining these repositories can help to understand software development, to support predictions about software development, and to plan various evolutionary aspects of software projects.

This chapter presents several analysis and visualization techniques to understand software evolution by exploiting the rich sources of artifacts that are available. Based on the data models that need to be developed to cover sources such as modification and bug reports we describe how to use a Release History Database for evolution analysis. For that we present approaches to analyse developer effort for particular software entities. Further we present change coupling analyses that can reveal hidden change dependencies among software entities. Finally, we show how to investigate architectural shortcomings over many releases and to identify trends in the evolution. Kiviat graphs can be effectively used to visualize such analysis results.

3.1 Introduction

Software evolution analysis is concerned with software changes, their causes, and their effects. It uses all sources of a software system to perform a retrospective analysis. Such data comprises the release history with all the source code and the change information, bug report data, and data that can be extracted from the running system. In particular the analysis of release and bug reporting data has gained importance because they store valuable information for analysing the evolution of software. While the recovery of the data residing in versioning systems such as CVS or Subversion has become a well explored topic, the ultimate challenge lies in the recovered data and its interpretation.

Some recent topics addressed in the field of analysing software repositories include the following:

T. Mens, S. Demeyer (eds.), *Software Evolution*.
DOI 10.1007/978-3-540-76440-3, © Springer 2008

- *Developer effort and social network analysis.* One of the goals in this topic is to find out the effort that team members are spending on maintaining and evolving software modules and how they communicate with each other. This allows a project manager to plan resources and reason about shortcomings in development processes and the team structure.
- *Change impact and propagation.* The main focus of this topic is to assess the impact of a change, such as the addition of a new or change of an existing feature, on the architecture, design and implementation of a software system. Being able to assess the impact of changes allows one to estimate the effort for maintenance and evolution tasks, to determine the impact of a change on the existing architecture and design of a system. Results are also used to provide guidelines for programmers such as if changing method `a` the programmer should also change method `b` and `c`.
- *Trend and hotspot analysis.* In this topic the trend of software entities is observed to find out shortcomings in the current architecture, design and implementation of software systems. Hotspots are the entities that frequently change and therefore are critical for the evolution of a system. One of the goals is to find heuristics and warning mechanisms that alarm project managers and architects of negative trends of software entities (and in particular of system hotspots) and provide suggestions to return the system into a stable state.
- *Fault and defect prediction.* A wealth of information is provided by software repositories that can be input to data mining and machine learning algorithms to characterize current and predict future properties of software entities. One property of software entities that is addressed by many approaches is the prediction of the location and number of defects in software entities such as source files. The result is a list of entities that will likely to be affected by defects which allows the development team to plan preventive actions such as refactoring.

In this chapter we address the first three topics and present techniques such as our Fractal Figures to analyse development effort, the Evolution Radar to analyse the change impact on source files, and Kiviat diagrams to analyse metric trends and to detect system hotspots. In the next section we present a general approach for analysing software repositories to understand software evolution. Examples of how to model and retrieve the data is presented in Section 3.3. The modeled data is the input for the different software evolution analysis techniques we present in Section 3.4.

3.2 An Overview of Software Repository Analysis

When mining software repositories one can consider many software artifacts: Source code from versioning systems, bugs from bug tracking systems, communication from e-mail lists or any further software artifacts such as documentation. This diversity of information is the foundation for many kinds of evolution analyses.