Improving Software Quality with Programming Patterns

Software systems and services are increasingly important, involving and improving the work and lives of billions people. However, software development is still human-intensive and error-prone. Established studies report that software failures cost the U.S. economy 60 billions annually and software vendors often spend 50-75 of the total development cost for finding and fixing bugs, i.e. subtle programming errors that cause software failures.

People rarely develop software from scratch, but frequently reuse existing software artifacts. In this dissertation, we focus on programming patterns, i.e. frequently reused code, and explore their potential for improving software quality. Specially, we develop techniques for recovering programming patterns and using them to find, fix, and prevent bugs more effectively.

We first developed Graph-based Object Usage Model (GROUM), a graph-based representation of source code. A GROUM abstracts a fragment of code as a graph representing its object usages. In a GROUM, nodes correspond to the function calls and control structures while edges capture control and data relationships between them. Based on GROUM, we developed a graph mining technique that could recover programming patterns of API usage and use them for detecting bugs. GROUM is also used to find similar bugs and recommend similar bug fixes.

Then, we developed Statistical Semantic LAnguage Model for Source Code (SLAMC). SLAMC represents code as sequences of semantic tokens for code elements like data types, variables, or functions. It implicitly captures programming idioms and patterns using a language model which incorporates code-based factors like local code context, global concerns, and pair-wise associations of code elements. Based on SLAMC, we developed a technique for recommending most likely next code sequences, which could improve programming productivity and might reduce the odds of programming errors.

Empirical evaluation shows that our approaches can detect meaningful programming patterns and anomalies that might cause bugs or maintenance issues, thus could improve software quality. In addition, our models have been successfully used for several other problems, from library adaptation, code migration, to bug fix generation. They also have several other potential applications, which we will explore in the future work.