Yunming Zhang's Blog

Research Direction Overview: Mining Source Code Repository

Posted on June 26, 2014

A research overview for mining source code repository based on a few seminal papers in the area.

It appears that there had been quite a bit of research in the area in the past decade, M.S.R(mining software repositories) has been the biggest workshop at ICSE and now a separate conference co-located with ICSE for 10 years. There has been some papers from PLDI that leverages code mining to solve certain problems.
There was a good overview paper on mining software repositories
"The Road Ahead for Mining Software Repositories"
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4659248&tag=1
and Tao Xie from UIUC actually maintained a bibliography of all the notable works in the field of software mining
https://sites.google.com/site/asergrp/dmse
A brief summary, it seems that
(1) There are three types of repositories that people mine for information on softwares, Historical repositories(svn logs.etc), Run-time repositories(deployment logs) and Code repositories(github, source forge, google code)
(2) Existing researches using the repository

1. **Reusing Code and assisting programming**
   A. locate uses of code such as library APIs, and attempt to match these uses to the needs of a developer **(this is the closest to what we discussed).** There are a large number of publications in this area.
   B. **Jungloid Mining: Helping to Navigate the API Jungle**, a well cited PLDI paper that uses repositories to infer type downcasts information that usually is not available statically. (I have a summary for this paper)
   C. A list of notable past works on assisting programminghttps://sites.google.com/site/asergrp/dmse/setasks#programming
2. Understanding Software Systems.
   A. Use history logs to understand rationale for certain unexpected designs in the code.
3. Propagating Changes
   A. changes to interface (Kathyrin's PLDI paper on systematic changes)
   B. automate change propagation can help avoid bugs
   C. code that change frequently together in the past are likely to change frequently in the future (historical repositories)
4. Predicting and Identifying Bugs
   A. best bug predicators are prior bugs and prior changes, i.e., chose that has bugs in the past is likely to have bugs in the future
   B. A list of works on static defect detection (bug detection) https://sites.google.com/site/asergrp/dmse/setasks#staticdefect
5. Understanding Team Dynamics
   A. Monitor and predict the productivity of a software engineering team through mails and IRC chats
6. Improving the User Experience
   A. prevent users perform actions that are reported to be "buggy" by other users

Some of these work uses code similarity, but a lot of them don't rely on code similarity measures. There is a good combination of data mining and programming language techniques.

Share this:

Related

Paper Summary:The Road Ahead for Mining Software Repositories
In "Programming Language"

Research Summary for Test Driven Code Reuse Tools
In "Programming Language"

Paper summary: for effective retrieval of examples in code repositories
In "Programming Language"

This entry was posted in Programming Language. Bookmark the permalink.