SCORE Project: Design Rationale Investigation Tool

INTRODUCTION

Developers working in a team environment must often work with code written by someone else, as when the code comes from a vendor, a developer is new to a team, a developer is working in unfamilar code for the sake of load balancing, or when debugging. Endeavoring to understand the rationale behind the implementation of such code is a common and difficult problem for developers. Developers need to understand such things as why the code was written the way it is, whether it's temporary code, how the code works, and what the code is trying to achieve [2].

Developers approach understanding the rationale behind code by reading the code directly, by examining its execution in the debugger or log files, reading bug reports, by examining the change log for the code [1]. Often these information sources are insufficient, and the developer needs to seek the advice either of the person who wrote the code or someone else on the team who may be familiar with it. Such "rationale investigations" are a difficult, tedious, and error-prone endeavor.

The purpose of this project is to build a tool to help developers do rationale investigations. The tool should synthesize as many relevant information sources as possible, including the source code, its comments, the bug/work item database, the source code control system change log, the team's or user's email and IM conversations, specs or design documents, the results of static analysis, the results of execution logs or dynamic analysis, etc. On the other hand the tool must not overwhelm the user, elegantly presenting the information most likely to be helpful but allowing the user to dig for details. Ideally the traces left by these investigations should be saved in a team-accessible repository and themselves become another information source.

APPLICATION DOMAIN AND SCENARIOS

Consider a team of 4-6 developers working closely together on a project, spread around the globe. The developers might be the entire software development staff of a small company, a team working on a subsystem of in a larger project in a medium- or large-sized company, or volunteer contributors to an open-source project. However it should be considered that each developer considers this project to be his or her primary work activity, making daily contributions to the evolution of the software.

A developer on the team arrives debugs his way into unfamiliar code. He can understand what the code is doing, but doesn't get why - it seems senseless to him. Moreover it is giving him results that he doesn't expect, so he needs to understand if the unfamiliar code is broken or if it is being invoked incorrectly. Since these are not apparent from direct examination of the code he invokes a tool that helps him investigate the vast amount of written and structured information at his disposal. He sees at a glance who wrote the code and gets a glimpse of the most salient features of the code's evolution. He looks in detail at the bug report that preceded the spec, which in turn preceded the initial implementation. He looks at the initial implementation and sees that it has the same mysterious characteristics, so he knows he should concentrate his investigation on the early history of the code. He finds an email conversation between the person who filed the bug and the person who wrote the spec, which gives him a crucial missing piece of the puzzle of the reason behind the code, enabling him to proceed with his debugging activity.

PROJECT GOALS (REQUIREMENTS)

1. Once configured, the system should work with little intervention.

2. The system must be reliable.

THE INTENDED OUTPUT OF THE PROCESS

1. The team may use any development process that they like, though must provide an initial plan identifying how they will approach the project.
2. An agile process is preferred, but a principled waterfall approach is also acceptable.
3. High-reliability is a requirement, so testing must be an integral part of the development
4. It is important that the UI be relevant to the customer, so the team should be principled about frequent informal UI testing and check-point meetings with the customer.

TOOLS AND STANDARDS

1. The team uses Microsoft Visual Studio Team System 2008 to manage their source code repository, their work items, and their bugs.

INTERACTION BETWEEN STAKEHOLDER AND DEVELOPING TEAMS

1. The team will provide monthly updates to the stakeholders.
2. The team and the stakeholders will maintain an open channel of email communication on an as-needed basis. The contact for the project is ginav@microsoft.com.
3. Microsoft will provide each selected team with a copy of Visual Studio Team System 2008 for their use on this project.
4. At most, two teams total will be selected to work on this project and/or its companion, "Awareness Tool for Distributed Software Team."

REFERENCES

[1] Andrew J. Ko, Robert DeLine, and Gina Venolia. Information Needs in Collocated Software Development Teams. In the Proceedings of the International Conference on Software Engineering (ICSE), May 2007.

[2] Thomas D. LaToza, Gina Venolia, and Robert DeLine. Maintaining mental models: A study of developer work habits. In Proceedings of the International Conference on Software Engineering (ICSE), May 2006.