Senior Conference Project Proposal

SketchyCode: An Accessible Interactive Thinking Space for Programmers.

Team 08: Z. Lockett-Streiff and N. Verosky

October 2, 2013

Abstract

We have identified two areas of hindrance in software development: the abstract representation of thoughts on easily disorganized physical media and information overload in IDE displays. We plan to design SketchyCode, an Android app that can interact with an IDE to streamline both aspects of the development process by linking high-level idea sketching and project visualization with actual implementation.

The ability to abstractly sketch out thoughts on physical media, such as paper and whiteboards is a boon for developers wanting to visualize their ideas. However, these disjoint thoughts can quickly become disorganized, and the connections between them may be lost to other developers. SketchyCode grants users the ability to not only organize their ideas in a single location, but also to draw connections between them to indicate related ideas.

SketchyCode can also interact with the Eclipse IDE to navigate and refactor blocks of code, enabling users to modularize their development process and avoid getting bogged down with implementing high-level conceptual shifts in project organization through tedious sequences of low-level changes (eg. cutting and pasting code between modules).

Motivation

Writing code is often a messy process. A typical workflow in an IDE like Eclipse contains a number of modules, including the following:

1. The text editor window
2. A package explorer
3. A project outline
4. Other auxilary functionality which may vary with the type of project being developed.

Eclipse has done well with organizing these modules to optimize space for the editor window. However, this high concentration of information may overwhelm the developer's ability to focus their thoughts. Parnin *et al* have responded to this problem with CodePad, a specialized tablet computer. CodePads are intended to serve as interactive spaces for maintaining concentration in programming environments by integrating high-level sketching into the development environment. The devices exist in multiple form factors, and can be linked with an IDE. The developer can use CodePads to navigate, refactor and annotate code in an external space.

While effective in reducing the extent of clutter in programming spaces, CodePad's user interface may still present the user with very dense amounts of information. The abstraction and fluidity inherent in the pencil-and-paper-guided sketching is lost. We intend to restore design abstraction in SketchyCode by providing an interactive space for users to brainstorm and visually organize their ideas. Additionally, SketchyCode will offer code navigation and reforactoring functionality. Containing the brainstorming process in a single location which still allows for abstract thinking and organization of disparate thoughts will streamline the design process for developers by granting them multiple levels of program design.

Furthermore, CodePad is designed to work on a single line of specialized hardware, which limits user accessibility to this interactive space. Such devices are likely to be very expensive. SketchyCode will begin as a free Android tablet application than can be generalized to other mobile devices.

Background

The primary impetus for SketchyCode is the work of Parnin *et al*, the developers of CodePad. CodePad provides peripheral working spaces which link to an IDE and enable developers to share notes and code with each other. The form factor on which we are modeling SketchyCode is the portable CodePad, a mid-sized tablet. However, the hardware of the portable CodePad is limited by the bulk of tablet computers leading up to and during 2010. The authors admit "a more apt device comes in the form of an iPad."[PGR10]. In the years following the SOFTVIS symposium, tablet form factors have been drastically streamlined. The authors are collectively more experienced with Android development, and so we will be creating our project on the Android platform.

The niche of our project is unique since mobile applications have only recently become popular. For reference, the first iteration of the iPhone was released in 2007, and the iPad not until 2012, two years after the CodePad paper was published. A cursory scan of Google Play revealed very little: an application titled "Android CodePad" which acts as a stand-alone code viewer. Android CodePad has no IDE integration or brainstorming space. This app is rated at a paltry 3.2 stars out of 5 likely bears no relation to the CodePad developed by Parnin *et al.*

Our Idea

To facilitate high-level code sketching on mobile devices, SketchyCode will represent coding projects as collections of free-floating modules on background canvases, enabling developers to quickly zoom into different modules for annotation, change spatial relationships between modules to conceptualize the project from different angles and allow for flexible birds-eye-view sketching, and refactor code by moving functions and classes between modules. We plan to implement SketchyCode as an Android application in communication with a back-end Eclipse plugin over the network so developers can conveniently move back and forth between SketchyCode for high-level sketching and reconceptualization, and Eclipse for more detailed writing and debugging of actual code.

Milestone Goals

We are breaking down our project as follows:

- **Milestone 1:** Mobile front-end coded up. Implementation of background canvas with free-floating modules and sketching capabilities complete, but SketchyCode can not yet communicate with Eclipse. Eclipse plug-in started.
- **Milestone 2:** User testing with dummy data completed. Progress on Eclipse plug-in.
- **Milestone 3:** Eclipse plug-in communicating with the Android app.

References

[PGR10] Chris Parnin, Carsten Gorg, and Spencer Rugaber. Codepade: Interactive spaces for maintaining concentration in programming environments. In *SOFTVIS*, 2010.