TeamLead Application Project Outline

James Williamson

EEC 626 – Software Engineering Project

Spring 2017

Revision History

Version	Author	Date	Description
0.1	James Williamson	01/23/2017	Original revision.
0.2	James Williamson	01/28/2017	Added project sponsor and tools information.

Document Conventions

In this document, “wish lists” are used to convey desired functionality beyond the bare minimum capabilities of the application. These pieces of functionality are keyed with the following symbols:

1. The functionality is straightforward, bounded in scope, and easy to implement.
2. The functionality is more complicated or open-ended, moderately difficult to implement.
3. The functionality is ambitious in scope and complex, more difficult to implement.

Table of Contents

1) Purpose

The purpose of this document is to describe the proposed functionality of the TeamLead application that will be developed as part of the EEC 626 Software Engineering Project at CSU.

2) Background

Leading a software development team brings with it several unique challenges. A software team lead must be adept at development, coaching/mentoring, architecture, release engineering, project planning, and more. This requires the ability to switch tasks and contexts easily, and proficiency in the analysis and management of very large amounts of information. Time management and organizational skills are paramount.

As such, the TeamLead application is proposed in order to aid software team leads by empowering them to leverage of one of today's most important and prevalent tools – the smartphone. There are myriad tools currently available for information management, communication, and note-taking, but most are either too generic or heavyweight to be consistently useful for the unique demands of this discipline. Worse, the learning curve may be too steep; this is an important consideration in a field where countless tools, frameworks, languages, and platforms must be adopted and utilized. TeamLead will provide a streamlined, user-friendly, and intuitive way for team leads to manage their time, capture and organize important information, and share content with developers.

3) Proposed Functionality

The TeamLead application will install on an Android smartphone and is divided into different pieces of functionality, described in the following sections.

3.1) Context Switch

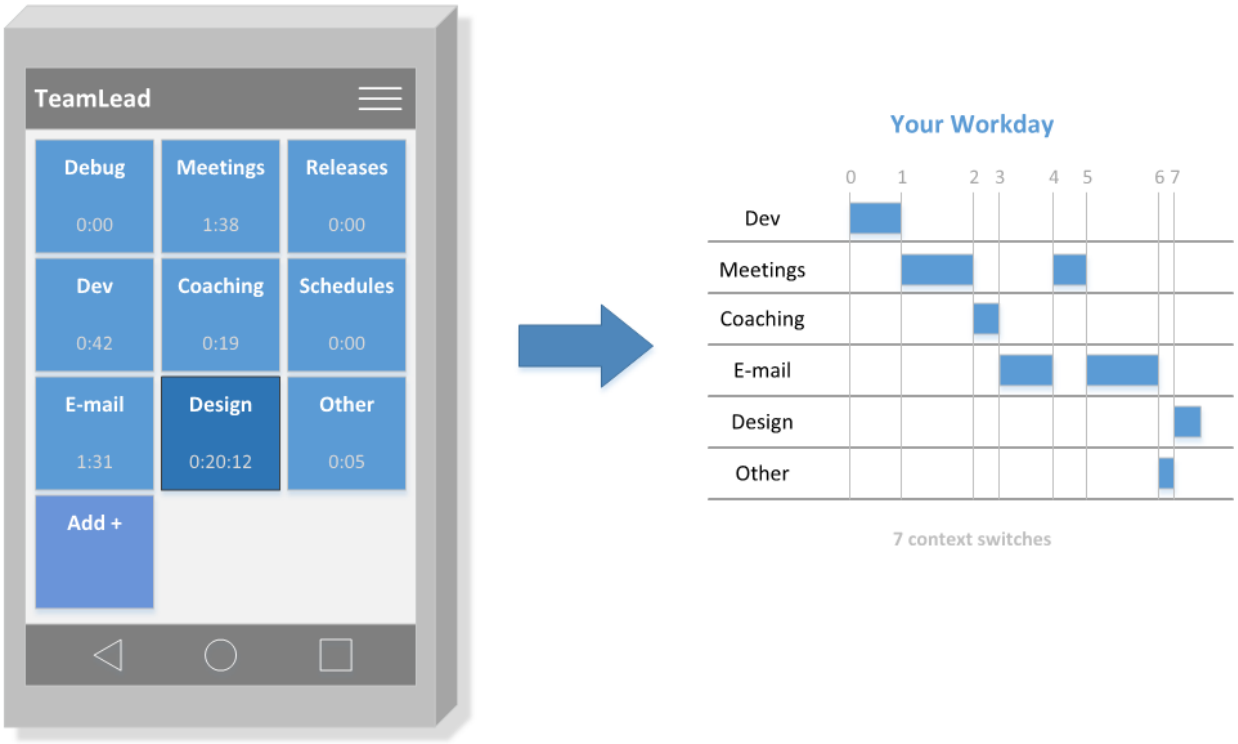
The "Context Switch" screen allows the user to record their current activity with a single tap. The phone display is divided into a grid of tiles, each representing a daily activity. These may include things like:

- Requirements elicitation
- Developing source code
- Writing documentation
- Debugging
- Release Engineering
- Design/Brainstorming
- Team Meetings
- Project Stakeholder Meetings
- Coaching/Mentoring
- Answering technical questions or e-mail

The user may additionally customize what is shown on the grid, or add their own tiles. When a tile is pressed, it indicates that the user has “context switched” to that activity and recording for the time spent on that activity will commence. The total time elapsed is shown as a small ticking timer on the tile. When a different tile is pressed, the timer for the current task stops, and time for the new activity is then recorded.

When the workday is complete, the user will hit the “done” tile, which will collate all data and present a chart depicting the activity breakdown (e.g. user spent 2:06 on requirements elicitation, 0:30 on coding, 1:32 on answering e-mail, etc.). This chart may then optionally be pushed to a website for logging and analysis.

Collecting this data allows the team lead to understand how much of their day is allocated to specific activities, helping them to tune their workday appropriately given current organizational demands. More importantly, the process requires at most two clicks (one to awaken phone, one to context switch when application is loaded) for typical operation, so it is lightweight enough to be useful even during hectic workdays.



Wish List:

1. Allow addition of custom tiles.
2. Allow robust tile reconfiguration and customization on GUI (grid density, colors, icons, etc.)
2. Allow notifications to be configured so as to grab a user's attention when too much time is being spent on one task.
2. Support multiple chart types for reporting time (pie chart, time-slice diagram, etc.)
3. Allow user to push time chart to the cloud for later tracking and analysis.
3. Allow context switching via voice recognition, in addition to standard touch input.

3.2) Tech Scrapbooking

"Tech Scrapbooking" is a function that enables a team lead to gather and annotate pieces of information on their phone in a fluid, intuitive way, so as to communicate important data to the team in a concise and logical manner. A "scrapbook entry" may consist of a mix of the following individual artifact types:

- Photograph
- Annotated photograph (such as a device or circuit board with arrows or text)
- Plaintext snippet
- Code snippet
- Voice recording
- Block diagram
- Requirement

Each "scrapbook entry" uses these artifacts to tell a story or convey instructions. Example entries may include:

- Instructions for bringing up a new piece of hardware
- Instructions for reproducing a difficult or transient anomaly
- Brainstorming sessions
- Information or diagnosis on a system failure
- Testing information

To use this function, the user first creates a "scrapbook entry" and gives it a name and, optionally, specific tags (used to sift through entries later). The user then uses the artifact collection function on the application to snap a photograph, write a note, record their voice, and so forth. With each gathered artifact, the user can directly assign it to a scrapbook, or leave it as unlinked/uncategorized "in the pile." Following this, a swipe upward sends the artifact to the cloud, whereas a swipe downward keeps the artifact on the phone.

Once the scrapbook is assembled and sent to the cloud, a developer may log in to view the artifacts. An example case of where this may be useful is depicted below.

"The *X-module* firmware team was informed that the *A-Z system* testing team reported an anomaly that only became apparent once the entire architecture was assembled and tested. It was a hard-to-reproduce timing issue that required a very specific sequence of steps. The visibility on this issue was very high, since a product launch was imminent. As such, the team lead went directly to the system testing lab to investigate.

Upon arrival, the team lead created a new scrapbook entry titled "Pre-launch system test anomaly investigation" and began collecting pieces of information:

- A block diagram of the system test setup/architecture
- Text notes on module numbers and versions of software
- A photo of the wiring with annotations on power lines, signal lines, etc.
- A pseudo-code snippet of the test algorithm

With each artifact, a single flick sent it to the cloud where developers back at their desks began to parse the information and use it to analyze the module codebase. Within the hour, the issue was found."

Wish List:

1. Allow L0 block diagramming artifact creation.
1. Allow basic photo annotation.
2. On "code snippet" artifact entry screen, display special keyboard with common language-agnostic keywords/constructs (if statements, loops, switch statements, etc.) for speedy data entry.
2. Allow L1 block diagramming artifact creation.
3. Allow swipe gesture functionality to send the artifact to the cloud ("up") or keep on the phone ("down"). Artifacts are chronologically sequenced in their scrapbook by default, but can be rearranged from the scrapbook viewer later.
3. Provide the capability to log into the website and reorganize/edit scrapbooks from the phone.

4) Tools

- **Toolchain:** Android Studio (https://developer.android.com/)
- **Test Emulator:** Various (supported by Android Studio)
- **Test Hardware:** Google Nexus 5 phone (personal)
- **Language:** Java

Web development tools to be determined at a later date for more advanced features.

5) Project Sponsors

Christopher Woggon (primary)
Senior Embedded Software Engineer
Rockwell Automation
cwoggon@ra.rockwell.com

James Williamson (self - secondary)
Senior Embedded Software Engineer
Rockwell Automation
jewilli1@ra.rockwell.com
https://www.linkedin.com/in/jameswilliamsonjr