A Novel Application of Open Source Technologies to Measure Agile Software Development Process

Luis Corral, Andrea Janes, Tadas Remencius,
Juri Strumpflohner, and Jelena Vlasenko

Free University of Bozen-Bolzano
Piazza Domenicani 3
39100 Bolzano-Bozen, Italy
luis.corral@stud-inf.unibz.it,
{andrea.janes,tadas.remencius,juri.strumpflohner,
jelena.vlasenko}@unibz.it

Abstract. In the last 10 years Open Source products have been widely used in industry. New methodologies and best practices to develop Open Source software appeared. In this work, we present an application that runs on Android-based mobile phones and collects proximity data with other devices via Bluetooth. The application gives new insights into measuring proximity inside a team of software developers. Data collection process is automatic so that the team members are not distracted from their daily activities. The collected data represent time frames when developers work alone at their machines and when they do Pair Programming with their colleagues.

1 Introduction

In the last 10 years Open Source based products have been widely used in industry [7], including platforms for mobile devices. For example, from the several operating systems for mobile phones (iOS, RIM OS, Windows Mobile, Android, etc), Open Source Android OS stands out as one of the leading platforms. Android is based on Linux, and ships with tools that provide significant support support for mobile-oriented software development. Moreover, Android OS supports many technologies, e.g. 3G, WiFi, Bluetooth, and many others, making it a powerful platform. Taking advantage of such technologies and its Open Source nature, it opens a new space for mobile software development. In this work, our focus is to take advantage of these characteristics and exploring how Open Source platforms and tools can be leveraged for measuring software development process.

Starting in the early ‘90s there have been several proposals for metrics for software processes and products [1, 3, 4, 12]. Since then, the way in which people develop software has changed dramatically. To determine how software is developed, there have been several studies that have been dedicated to automatic and non-invasive observation of behavior of software developers [3], in terms of code quality [9], time to market, effort distribution [1], knowledge transfer, etc.

I. Hammouda et al. (Eds.): OSS 2012, IFIP AICT 378, pp. 316–321, 2012.
© IFIP International Federation for Information Processing 2012

The purpose of our research is to take advantage of such experience and to propose a new set of tools and methodologies to collect and interpret software metrics coming from emerging software production environments [8], making use of technologies and possibilities that mobile resources offer. In this paper, we present DroidSense, a mobile application for Android based smartphones, that utilizes Bluetooth technology to measure proximity between team members. Using these data we can detect when developers work alone and when they do Pair Programming, working collaboratively on one task using a single machine.

2 Related Work

There have been several works aiming to compare existing mobile operating systems in terms of different quality characteristics. In [2], authors evaluate architectural openness of iPhone, Windows Mobile, Android, and others. The results evidence that Android and Symbian are more open for modifications than other platforms, making them attractive to carry out research and experimentation.

In [13], authors explore Bluetooth potentials for designing interacting systems. Several benefits and weaknesses of this technology have been identified, for example, Bluetooth chips can be carried by a person or placed in a specific location. Device names can be easily changed to uniquely identify persons or groups. With this, from the obtained data it is always possible to detect what kind of device has been encountered. However, Bluetooth is limited by short-range radio technology, so that only nearby located devices can be detected. Several scans might be needed to identify all the participating devices. For product assessment, in [11] authors developed BlueMonarch, an application to evaluate Bluetooth-based applications.

In this study, we take advantage of such technologies and to apply them for software metrics collection. There is also a significant number of devices supporting Bluetooth communications: phones, headsets, hand-helds, laptops, etc. Some of such devices, like phones and headsets, very often are carried by people wherever they go, thus, the presence of such devices can provide a detailed map of the physical activities that developers do during their work, meetings, etc. Tools using Bluetooth to determine proximity exist, like ReduxComputing-Proximity [5], Where's Blue [6], and Proximity Data Gathering For Android Through Bluetooth [10]. However, they are primitive and not yet adopted to extract detailed software metrics.

3 DroidSense

DroidSense is a system for the automated collection of proximity information between developers for software process analysis. DroidSense consists of a mobile Android application and a central server component responsible for receiving, storing, and analyzing the data. The collected data can then be viewed and analyzed through an according web interface providing different kinds of visualizations. The structure of the system is shown in Figure 1:

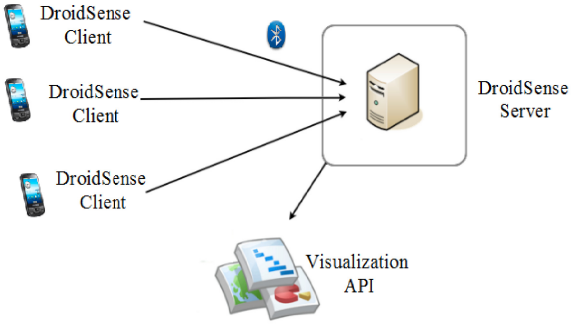

Fig. 1. High Level System Architecture

DroidSense Android client is the core component of the system, being responsible for the data collection. It has been designed to be easily extensible to allow the development of future Android activities that are able to attach to its functionality. The end-user mainly interacts with the Android client that collects and uploads the data to the server. The main user interface presented to the user when he starts the application is the list showing all of the discovered Bluetooth devices.

By selecting an entry of the device list, the according device detail view opens. In addition to the device's detail information, it presents the collected proximity raw data in descending order, showing the most recent value on top of the list. Each entry shows the calculated value on top, followed by the raw value that has been returned from the Bluetooth adapter as well as the time stamp of the discovery.

DroidSense server, on the other hand, is responsible of storing and analyzing the collected values by all of the DroidSense Android clients. It has been implemented by using a conjunction of Open Source-only application frameworks, on top of J2EE and Spring. Spring supports Hibernate, which has been used to perform the object-relational mapping to the underlying MySQL database. Open source technologies JDOM and Castor prepare the XML received from DroidSense services into Java objects that can be understood by JDBC to communicate with the database instance. For the web front-end, the Spring Web MVC framework has been used with JSP and jQuery.

4 DroidSense in Action

4.1 Case Study

To gain experience on the collection of data using our tool, as well as to prove their practical usability for the purpose of software process measurement, it was conducted an experiment of the use of DroidSense by a professional team. An initial implementation of the prototype took place on a team of 10 programmers, members of a software development team in a University Research Center which uses Agile practices such as Extreme Programming, Pair Programing, and Test-Driven Development. All participants joined the experiment in a voluntary basis.

The experiment covered a period of 3 months. Through the whole phase, more than 240,000 proximity values have been transmitted by the 10 participants. The samples were analyzed with the purpose of reconstructing the whole -or at least significant parts- of the software development process, based on human interaction. For privacy reasons, all personal names have been obfuscated.

4.2 Interpretation of Data Samples

An example on how data are interpreted is depicted in Figure 2, which shows a visualization of the raw data collected on a regular working day, from the viewpoint of developer U_E at the University, our experimental setup. The y-axis shows the RSSI indicator, where lower values indicate shorter distances. The x-axis shows the time of the day in hours.

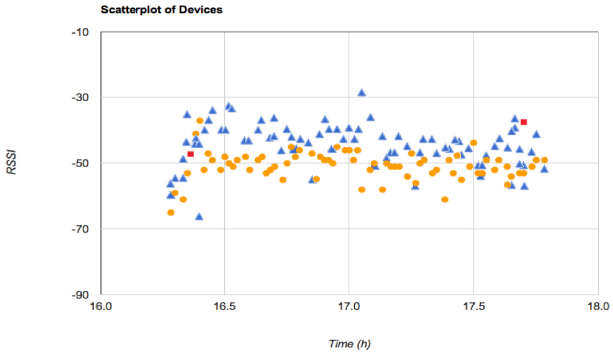

Fig. 2. Raw Values Collected at the University Research Center

The visualization contains data of the devices:

- P_E: U_E's personal laptop, used at work (triangles)
- M_M: DroidSense-equipped Android phone of a member of U_E's team (points)
- M_S: Android phone with DroidSense, member of U_E's team (squares)

Figure 3 shows an interpretation of the data acquired by DroidSense, saying that M_M was near to P_E during the whole period of the scan, while M_S dropped in for a small amount of time at the beginning and the end of the scan (as seen in Figure 2). A pair programming session between user U_E and M_M can be inferred:

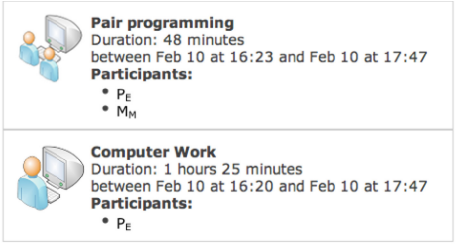

Fig. 3. Calculated Sessions at the University Research Center

On the other hand, another pair programming session took place in front of the workstation of U_E itself, while the pair programming session shown previously was in front of the teammate's computer. The output is not presented as a list of detailed session detections, but rather the aggregated information of all of them. This helps on giving an overview of the activities performed through the day. It can be seen that the raw data is correctly interpreted, showing the user M_S only appear for a small amount of 5 minutes during the scan period, while there is a longer pair programming session with M_M.

Project's team managers at the Research Center reported a clearer identification of the execution of pair programming sessions, recognizing how devices group in front of a particular workstation for a given amount of time. DroidSense was useful to understand the interaction among developers: the distance between users and devices can be appraised, allowing to observe how such distance shifts when one user approaches a colleague, leaving his own workstation. This visualization was reported by the team managers as suitable to conduct further analysis on the importance of people's physical distribution in the way a team collaborates.

5 Conclusions

Measurement has become a major factor of importance in software development. Project deadlines, limited budget and high customer expectations require the continuous search for optimizing existing processes.

In this work we presented a new approach in collecting data about Agile software development process by leveraging proximity measurements gathered using the Bluetooth technology through DroidSense, a system for Android OS which implements a software solution to characterize developer's interaction using data gathered from their own mobile devices. Thanks to an experiment with several developers conducting an Agile software development project, we present the practical use of DroidSense and the resulting process analysis. Using DroidSense shows that the collected data provides a valuable insight in the stakeholder's interactions within the process by automatically detecting a developer's involvement in computer work or pair programming sessions.

Using a widely spread technology such as Bluetooth, an Open Source operating platform like Android, and a wide range of Open Source frameworks and tools allows a team to slim down the development investment without losing the functionality required by architecture, design and implementation demands. Moreover, there is no costly setup required but installing a mobile app, and engineers do not have to undergo a major training phase or adapt their working habits.

References

1. Abrahamsson, P., Moser, R., Pedrycz, W., Sillitti, A., Succi, G.: Effort prediction in iterative software development processes–incremental versus global prediction models. In: Empirical Software Engineering and Measurement (2007)

2. Anvaari, M., Jansen, S.: Evaluating architectural openness in mobile software platforms. In: ECSA 2010, Copenhagen, Denmark (2010)
3. Coman, I.D., Sillitti, A., Succi, G.: A case-study on using an automated in-process software engineering measurement and analysis system in an industrial environment. In: ICSE 2009, Vancouver, Canada (2009)
4. Fenton, N.: New directions for software metrics. In: PROMISE Workshop, Invited Keynote, ICSE 2007, Minneapolis, USA (2007)
5. Google, `http://code.google.com/p/reduxcomputing-proximity` (retrieved March 10, 2012)
6. Engelsma, J.R., Ferrans, J.C., Hans, M.C.: EncounterEngine: Integrating Bluetooth user proximity data into social applications. In: IEEE International Conference on Wireless and Mobile Computing, WIMOB 2008, pp. 502–507 (2008)
7. Gurbani, V.K., Garvert, A., Herbsleb, J.D.: A case study of open source tools and practices in a commercial setting. In: Proceedings of the 5th Workshop on Open Source Software Engineering (2005)
8. Janes, A., Scotto, A., Pedrycz, W., Russo, B., Stefanovic, M., Succi, G.: Identification of defect-prone classes in telecommunication software systems using design metrics. Information Sciences 176(24) (2006)
9. Moser, R., Sillitti, A., Abrahamsson, P., Succi, G.: Does refactoring improve reusability? Reuse of Off-the-Shelf Components, 287–297 (2006)
10. Pullabhatla, A., Gomes, H.: Proximity data Gathering for Android through Bluetooth. Department of Computer Science Engineering. University of California, San Diego (2010)
11. Smith, T.J., Saroiu, S., Wolman, A.: BlueMonarch: A System for evaluating Bluetooth applications in the wild. In: MobiSys 2009, Kraków, Poland (2009)
12. Succi, G., Pedrycz, W., Liu, E., Yip, J.: Package-oriented software engineering: a generic architecture. IT Professional 3(2), 29–36 (2001)
13. Sundstrom, P., Taylor, A.S., O'Hara, K.: Sketching in Software and Hardware Bluetooth as a Design Material. In: Mobile HCI 2011, Stockholm, Sweden (2011)