Adapting Linux for Mobile Platforms: An Empirical Study of Android

Foutse Khomh, Hao Yuan, Ying Zou
Department of Electrical and Computer Engineering
Queen's University
Kingston, ON, Canada
{*foutse.khomh, hao.yuan, ying.zou*}*@queensu.ca*

***Abstract*—To deliver a high quality software system in a short release cycle time, many software organizations chose to reuse existing mature software systems. Google has adapted one of the most reused computer operating systems (*i.e.*, Linux) into an operating system for mobile devices (*i.e.*, Android). The Android mobile operating system has become one of the most popular adaptations of the Linux kernel with approximately 60 millions new mobile devices running Android each year. Despite many studies on Linux, none have investigated the challenges and benefits of reusing and adapting the Linux kernel to mobile platforms. In this paper, we conduct an empirical study to understand how Android adapts the Linux kernel. Using software repositories from Linux and Android, we assess the effort needed to reuse and adapt the Linux kernel into Android. Results show that (1) only 0.7% of files from the Linux kernel are modified when reused for a mobile platform; (2) only 5% of Android files are affected by the merging of changes on files from the Linux repository to the Android repository; and (3) 95% of bugs experienced by users of the Android kernel are fixed in the Linux kernel repository. These results can help development teams to better plan software adaptations.**

***Keywords*-Empirical study; Software adaptation; Linux; Android; Operating system; bugs.**

I. Introduction

The flexibility of the Linux operating system has led to its adoption in a wide range of domains, from home PCs, enterprise desktops, servers to mainframes and supercomputers. With the increasing demand for mobile devices that provide as rich functionalities as desktop computers, many mobile carriers are also adapting Linux into mobile operating systems. Over the past five years, the deployment of Linux in mobile operating systems has risen to 60 million units per year, from different handset vendors, such as Motorola, NEC, Panasonic, and Samsung [1]. Software adaptation typically occurs when the cost of adapting an existing software to a new platform is estimated to be less than the cost of building a completely new software from scratch. During software adaptation, three principal phases are particularly important: (1) the initial modification of the existing software system (*i.e.*, the original system) to reuse its functionalities in the adapted system; (2) the maintenance of the adapted system to keep it updated with changes from the original system; and (3) the corrective maintenance of the adapted system to fix bugs carried down from the original system as well as the new bugs introduced during the previous development phases of the adapted system. These three phases can be more expensive than expected if the software adaptation is not done properly. For example, if the design of the original system is not well understood by the developers carrying the adaptation, they can perform erroneous changes invalidating key design decisions and causing the maintenance of the adapted system to become very expensive. A great amount of research has investigated the development of Linux (*e.g.*, [2], [3]). The Android system built from the kernel of Linux 2.6 has also been analyzed extensively (*e.g.*, [4], [5]), yet, to the best of our knowledge, no study has investigated the challenges and benefits of Linux adaptations, in particular, Linux adaptations to mobile platforms. A good understanding of the challenges and benefits of Linux adaptations would be beneficial not only for the Linux community, but also for practitioners who may be interested in adapting the Linux system to new platforms. In this paper, using software repositories from the Linux and Android systems, we aim at assessing the ease to adapt the Linux kernel into the mobile operating system Android and the benefits of adapting from the Linux kernel instead of developing a new operating system from scratch.

This work makes the following contributions:

- We assess the initial effort needed to adapt the Linux kernel into Android and found that 99% of the functionalities of Linux kernel 2.6 were reused into Android, and only 0.7% of the reused files were modified to implement the requirements of Android.
- We measure the effort required to maintain the kernel of Android, and found that in average only 5% of Android files are affected by the merging of changes from the Linux repository.
- We also investigate the propagation of bugs from Linux to Android and observed that 95% of bugs experienced by users of the Android kernel come from Linux and these bugs are fixed by Linux contributors.

The rest of this paper is organized as follows: Section II introduces some related works on Linux and Android systems. Section III describes the design of our study. Section IV presents and discusses our results. Finally, Section V concludes the paper, discusses some limitations and outlines avenues for future work.

978-1-4673-2312-3/12/$31.00

II. Related Work

This section presents some works on Linux and Android.

Linux System: Linux is one of the most studied software system in the literature. Godfrey and Qiang [2] have explored the evolution of the Linux kernel both at the system level and within the major modules and found its growth rate to be super-linear. They also found that even though the entire source code of Linux is quite large, more than half of the code consists of device drivers, which are relatively independent of each other. Israeli and Feitelson [3] analyzed 810 releases of the Linux kernel over a period of 14 years and observed a decrease of the average complexity.

Android System: Major studies on Android have focused on its reliability. Maji *et al.* [5] compared the reliability of Android and Symbian OS through an analysis of reported bugs. They concluded that more than 90% of bugs in both Android and Symbian are permanent in nature, with 22% of Android bugs requiring major code changes. Gronli *et al.* [4] compared the development environments of Android, Windows mobile, and Java ME, and conclude that Android and Window mobile have better environments. We investigate the adaptation of the Linux kernel into Android.

III. Study Design

The *goal* of this study is to investigate the challenges and benefits of adapting the Linux kernel for mobile operating systems. The *perspective* is of practitioners who are interested in adapting the Linux kernel to new platforms, in particular, mobile devices. The Linux system is one of the fastest growing operating systems used in a variety of different platforms. Moreover, the results of this study can be of interest to Linux developers who could take into consideration some of the challenges and benefits of adapting Linux during their development activities. Finally, the results of our study can be of interest to researchers interested in adapting large legacy systems onto new platforms.

The *context* of this study consists of the Android system. We chose this system for three reasons. First, the Android system is one of the most successful mobile systems using a modified Linux kernel in the current market. It is a good representation of the adaptation of Linux for mobile devices. Second, the Android system is a mature commercial product. The software contains all the necessary functionalities of a mobile operating system. Last but not least, it is open source; we have access to all the repositories of the Android system. These criteria make Android a very suitable candidate for our study. We analyze 6 versions of Android developed between 2005 and 2010. We investigate development efforts during the following phases:

1) The adaptation of Linux functionalities into Android.
2) The maintenance to keep Android updated with changes from Linux.
3) The corrective maintenance to fix bugs carried down from Linux as well as new Android's bugs.

A. Data Extraction

Both Android and Linux use the source code management system Git[1], to track modifications on their respective source code. We extract log files from both Linux and Android Git repositories. Because commits in Git repositories come from multiple time zones, we convert the time zones of the extracted logs to the Eastern Time chosen as our reference. Next, we extract the following information from each commit log: commit ID; the list of committed files; the number of lines of code added or deleted; and the kind of the commit, *i.e.*, merging commits, or bug-fix.

Git allows developers to annotate each modification of the source code with a message describing the reason for the modification. We parse such messages using a perl script to extract references to bug reports such as URLs from bugzilla or bug Ids, (*e.g.*, http://bugzilla.kernel.org/show_bug.cgi?id=5209). We also look for specific keywords, such as "fixed" or "bug" in a way similar to the work by Sliwersky *et al.* [6]. This approach enables us to track Linux bug fixes as well as merging commits in the Android kernel. We rely on the code review system of Android (*i.e.*, Gerrit[2]), to track bugs related to files existing only in the Android, *i.e.*, bugs that have no root in Linux. We parse messages describing change reviews in Gerrit to search for keywords such as "fixed" or "bug" in Git commit messages. We perform a syntactical analysis to retrieve information about modules and packages that were changed. In total, we identified 1,058 bug fixes. To assess the precision of our bug fix identification, we randomly selected 200 classified instances, and manually verified them. We found a precision of 86%.

IV. Study Results

This section presents and discusses the results of our analysis of the adaptation of Linux kernel 2.6 into Android.

A. Adaptation of Linux Functionalities into Android

1) Motivation: We want to quantify the effort required to adapt the kernel of Linux into a mobile operating system. Decision makers in software organizations could use this quantification when planning software adaptations.

2) Analysis Method: We analyze the source code repositories of Android and Linux kernels. More specifically, we analyze the similarity between the two systems, in terms of their functionalities and the amount of changes that were performed to integrate Linux functionalities into Android. To measure the level of functionality reuse between Linux kernel 2.6 and Android, we introduce the $Similarity$ metric defined in Equation (1).

$$Similarity(A,B) = \frac{|Source_A \bigcap Source_B|}{|Source_A \bigcup Source_B|} \quad (1)$$

[1] git-scm.com/

[2] http://code.google.com/p/gerrit/

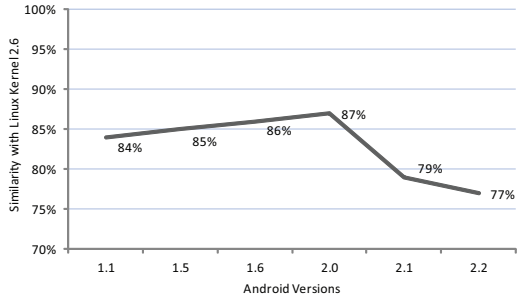

Figure 1. Evolution of the Similarity between Android and Linux Kernel

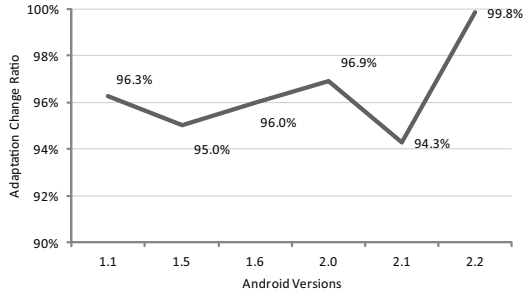

Figure 2. Percentage of Adaptation Changes Overtime

$Sources_S$ is the set of source code lines from the system S. $Similarity(A, B)$ is 1 when systems A and B are identical.

To measure the amount of changes performed by Android developers to adapt Linux functionalities, we introduce the Adaptation Change Ratio (ACR) defined in Equation (2).

$$ACR = \frac{Number\ of\ non-merging\ commits}{Total\ number\ of\ commits} \quad (2)$$

3) Results: We found an 84% similarity between the source code of Android 1.1 and Linux kernel 2.6. Moreover, 99% of Linux kernel functionalities are reused into Android and only the file system and memory management modules have been modified. In total, Android added 89 new files and modified 75 Linux files, which represents 0.7% of all Linux files. During this initial adaptation phase, Android developers made 3,741 commits before the first release, *i.e.*, Android 1.1. The Adaptation Change Ratio (*i.e.*, ACR) was 0.963; meaning that 96.3% of commits during the initial adaptation phase were modifications to adapt and extend Linux to the mobile platform. Only a very small (*i.e.*, 3.7%) portion of commits were updates from the Linux repository.

Figures 1 shows the evolution of the similarity between the Linux kernel and Android overtime. From Android 1.1 to Android 2.0, the similarity between the two systems increased steadily. This observation can be explained by Android importing new features and bug fixes from the Linux kernel. However, after Android 2.0 the similarity between the two systems is decreased. This decrease is caused by the large amount of new functionalities added to Android 2.0. Looking at the evolution of the Adaptation Change Ratio, shown in Figure 2, we observe a sharp increase (*i.e.*, up to 99.8% of commits) after the release of Android 2.1; meaning that developers had to perform more changes to integrate the newly added functionalities with the functionalities imported from the Linux kernel.

We investigated the proportion of adaptation changes on each module of Android, and found that the most modified modules are *drivers* (*i.e.*, 31%) and *fs* (*i.e.*, 25%). The distribution of changes on remaining modules is: *include* (17%), *arch* (11%), *net* (7%), *kernel* (6%), and *mm*(3%).

B. Maintenance to Update Android with Linux's Changes

1) Motivation: In a traditional software development process, maintenance is estimated to cost between 50% and up to 90% of the total cost of the software system [7]. Adapting an existing software to build a new software can introduce challenges that are not generally faced in a traditional development process. In this section we quantify the amount of changes needed to keep Android up to date with new features and bug fixes from Linux. Such results would provide development teams with a better understanding of the consequences of adapting a system like Linux.

2) Analysis Method: During maintenance, Android developers perform periodical source code merges from the Linux code repository. To measure the effort induced by these merges, we compute the Density of Merging Commits (DMC) metric following Equation (3).

$$DMC = \frac{Number\ of\ merging\ commits}{Total\ number\ of\ commits} \quad (3)$$

DMC captures the effort spent on merging source code modifications from the Linux repository.

Very often, because of data or control dependencies, a source code merge from Linux to Android will result in a series of cascading changes on other parts of Android. To capture this impact of a merge, we introduce the Impact of Merging Commits (IMC) metric defined in Equation (4):

$$IMC(i) = \frac{B_i}{M_i} \quad (4)$$

Where i is a merging commit; B_i is the number of Android's files that were modified because of dependencies with files contained in i; and M_i is the number of files contained in the merging commit i.

$IMC(i)$ computes the proportion of files that were updated to solve conflicts caused by a the merging commit i.

Because it is easier to update files from a single module compared to files distributed across multiple modules, we introduce the Impact Range of Merging (IRM) metric, defined in Equation (5), to calculate the number of modules affected by a merging commit:

$$IRM(i) = Impacted\ modules_i \quad (5)$$

Where i is a merging commit and $Impacted\ modules_i$ is the number of modules containing files that were updated to solve conflicts caused by the merging commiti.

We also compute the lasting impact of merges in terms of the number of sub-sequent commits triggered by a merging

commit. For each merging commit i, we count the number of subsequent commits containing files from merging commit i. We stop counting, when we find a sub-sequent commit with none of the files from the merging commit i.

3) Results: We found the DMC of Android to be 0.14, meaning that 14% of all Android's commits are merging commits to keep Android up to date with the latest modifications of the source code of Linux kernel. These merging commits represent 6.7% of all commits in the Linux kernel repository. Consequently, only small updates from Linux kernel are necessary to keep Android effective.

On average, only 5% of Android files are modified as a result of a merge. This percentage is low compared to the percentage of Android files that are modified after a feature enhancement or bug-fix commit (*i.e.*, 96%). However, during the period from 2009 to 2010, we found few merging commits that required developers to revise all the merged files. We refer to them as "breaking" commits. We manually analyzed "breaking" commits and found them to be small in size; they contain between 3 and 20 files, while the average number of files in a merging commit is 374 files. Nevertheless, these "breaking" commits concern changes to important features of Linux, such as network features, or bug fixes. 84% of merging commits impact only 1 or 2 modules. The most impacted module is the kernel, with 35% of all merging commits causing a modification of the kernel of Android. Also, 86% of all merging commits have no lasting effect, *i.e.*, no subsequent commit is required to resolve conflicts caused by the merge. For the remaining 14%, a maximum of three subsequent non-merging commits were required to resolves issues caused by the merges.

C. Corrective Maintenance to Fix Bugs from Linux

1) Motivation: When adapting an existing software to build a new software, development teams have to deal with bugs from the original software. Bug fixes from the original software need to be propagated to the adapted software. At the same time, new bugs introduced after the adaptation of the original software need to be handled properly. In this section, we investigate how bugs imported from the Linux kernel and new Android's kernel bugs are fixed.

2) Analysis Method: Using the Bug Fixed Ratio (BFR) metric defined in Equation (6), we compute the proportion of Android kernel's bugs that are fixed by Android's developers. We expect that bugs declared in Android but caused by Linux code will be fixed by Linux's developers.

$$BFR = \frac{Android\ Fixes}{Linux\ Fixes} \tag{6}$$

Where $Android\ Fixes$ is the number of bugs declared in Android and fixed by Android developers and $Linux\ Fixes$ is the number of bugs declared in Android but fixed by Linux developers.

3) Results: In total, 423 files underwent bug fixes. Among them, only 13 files (*i.e.*, 3%) were corrected because of bugs that are specific to Android features (*i.e.*, these files were not imported from the Linux kernel). The BFR of Android is $\frac{1}{20}$, meaning that 95% of bugs are fixed by Linux's developers and only 5% by Android's developers. We investigated the reappearance of Linux bugs in Android and found none. Therefore, it may be a good idea to build new software systems from seasoned mature systems, since developers from the original system are likely to handle a significant part of the effort to maintain the adapted system.

We also analyzed the possible impact of Android's changes on the Linux kernel and observed no merging commits from Android to Linux. This finding is consistent with previous comments from the literature about the non-contribution of Android developers to Linux[3].

V. Conclusion

In this paper, we have analyzed the adaptation of the Linux kernel into a mobile operating system, *i.e.*, Android. Results show that 99% of Linux kernel's functionalities were reused into Android and only 0.7% of Linux kernel's files were modified during this adaptation. On average, only 5% of Android files were modified as a result of a merge from Linux and the lasting impact of a merging commits is less than 3 subsequent commits. 95% of bugs reported on the Android kernel are fixed by Linux's developers. Development teams should consider adapting Linux when possible, since there is a good chance that most of the maintenance effort of the adapted system will be lifted by Linux's development team. The metrics proposed in this paper are computed at a file level. In future work, we plan to extend our investigation at a lower level of granularity, *e.g.*, the code level.

References

[1] B. Weinberg, "Uniting mobile linux application platforms," *Linux Pundit*, 2008.

[2] M. Godfrey and Q. Tu, "Evolution in open source software: a case study," in *Software Maintenance, 2000. Proceedings. International Conference on*, 2000, pp. 131 –142.

[3] A. Israeli and D. G. Feitelson, "The linux kernel as a case study in software evolution," *J. Syst. Softw.*, vol. 83, pp. 485–501, March 2010.

[4] T.-M. Grønli, J. Hansen, and G. Ghinea, "Android vs windows mobile vs java me: a comparative study of mobile development environments," in *Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments*, ser. PETRA '10, 2010, pp. 45:1–45:8.

[5] A. K. Maji, K. Hao, S. Sultana, and S. Bagchi, "Characterizing failures in mobile oses: A case study with android and symbian," 2010.

[6] J. Śliwerski, T. Zimmermann, and A. Zeller, "When do changes induce fixes?" *SIGSOFT Softw. Eng. Notes*, vol. 30, pp. 1–5, May 2005.

[7] L. Erlikh, "Leveraging legacy system dollars for e-business," *IT Professional*, vol. 2, no. 3, pp. 17 –23, may/jun 2000.

[3] http://www.kroah.com/log/linux/android-kernel-problems.html?seemore=y