On Ad Library Updates in Android Apps

Israel J. Mojica Ruiz*, Meiyappan Nagappan*, Bram Adams§, Thorsten Berger†, Steffen Dienst‡, Ahmed E. Hassan*

*Software Analysis and Intelligence Lab (SAIL)
School of Computing, Queen's University, Canada

§Lab on Maintenance, Construction and Intelligence
École Polytechnique de Montréal, Canada

†Generative Software Development Lab
University of Waterloo, Canada

‡Chair of Business Information Systems
University of Leipzig, Germany

mojica,mei@cs.queensu.ca*, bram.adams@polymtl.ca§,
tberger@gsd.uwaterloo.ca†, sdienst@informatik.uni-leipzig.de‡,
ahmed@cs.queensu.ca*

Abstract

With more than 90% of mobile apps today being free-to-download, advertisement within apps is one of the key business models to generate revenue. Advertisements are served through the embedding of specialized code, i.e., ad libraries. Unlike other types of libraries, developers cannot ignore new versions of the embedded ad libraries or new ad libraries without risking a loss in revenue. However, updating ad libraries also has expenses, which can become a major problem as ad library updates are becoming more prevalent in mobile apps.

Hence in this paper, we first discuss the various expenses involved in updating ad libraries, then empirically explore the prevalence of "ad library updates" within Android apps. An analysis of 13,983 versions of 5,937 Android apps collected over 12 months shows that almost half (48.98%) of the studied versions had an ad library update (i.e., ad library was added, removed, or updated). Interestingly, in 13.75% of app updates (new version in the Google Play store) with at least one case of ad library update, we found no changes to the app's own API, which suggests substantial additional effort for developers to maintain ad libraries. We also explore the rationales for why such updates are carried out.

Keywords – Mobile apps, Advertisement libraries, Software maintenance

1 Introduction

Mobile apps are software applications developed for mobile devices, such as smartphones and tablets. Apps are available via online distribution channels called *app stores*, such as the Apple App Store, and Google Play. Across these stores, the mobile app market grew from $18 billion in 2012 to $26 billion of revenue in 2013. However, approximately 91% of the apps in such app stores are free-to-download [10]. Hence, many app developers use non-traditional revenue models to monetize their apps.

One of the most popular revenue models is the advertisement (ad) model, where ads are displayed within an app, and developers earn money every time users click on such ads. To serve ads, developers must embed specialized libraries – *ad libraries* – in their apps. Previous studies estimated that around 51–60% of free Android apps have at least one ad or analytics library [4, 7, 9]. Gartner projects that the revenue from mobile ads (both mobile apps and websites) will reach $18 billion in 2014 [18]. App Annie and IDC found that revenue from in app advertising has gone up by 56% in 2013 [2].

Given the highly competitive nature of the ad industry, ad libraries are updated on a regular basis to include new ad serving models, or for legal and financial reasons (e.g., merging of advertising companies). In that case, app developers need to decide if they wish to integrate the updated libraries into their app. While an ad library would not usually affect the core functionality of an app (hence most library updates can be safely ignored), such an update to an ad library might have a big impact on the revenue of an app. However, releasing a new version of the app primarily because of an updated ad library could be a hard sell (figuratively) to existing users. It costs money to download the new update (especially on 3G) and users also need to worry about whether it would be safe to update to the new version, i.e., does it break existing behaviour or corrupt existing data?

Thus, updates to ad libraries could be expensive directly in terms of maintenance effort, and indirectly in terms of lost revenue when skipped. More details about these various expenses related to ad library updates in mobile apps are presented in Section 3. If such updates are prevalent among mobile apps, then the software engineering research community and the ad library companies need to come up with innovative solutions to reduce these expenses. On the other hand, if such updates are not prevalent or solutions could be found to reduce the associated expenses of ad library updates, then these expenses may not be that costly.

However, there is no current research that examines the prevalence of ad library updates in mobile apps. Hence, the main contribution of this article is to empirically investigate the prevalence of ad library updates in mobile apps by mining 13,983 versions of 5,937 Android apps over 12 months (Section 4). We find that almost half (48.98%) of the studied versions of Android apps had an ad library that was added, removed, or updated (for convenience, we call all of these cases, "ad library updates"). In Section 6, we dig further, based on our results in Section 5, to determine why app developers needed to update the ad libraries in their apps. Finally, we conclude in Section 7.

2 Background and Related Work

In this section, we present some background information on ad serving models, and introduce key terminology with respect to ads.

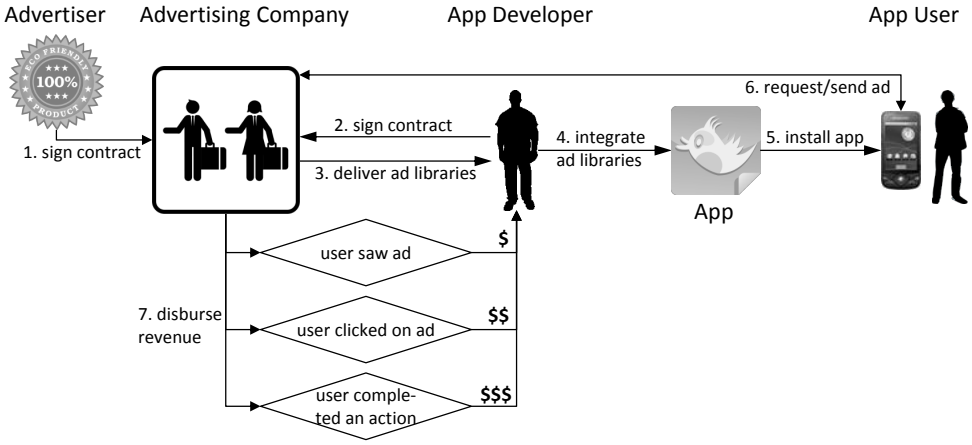


Figure 1: The process for serving ads in mobile apps.

2.1 Ad Serving Models

Figure 1 gives a simplified overview of the process for serving ads. The entity that wants to put mobile advertisements – the advertiser – signs a contract with an advertising company. The advertising company provides ad-libraries that app developers can integrate into their apps. These libraries are invoked

at run-time in order to serve ads based on communicated information about the users (e.g., their location, or their interest). Developers receive financial compensation when users interact with the ad, for example, when a user sees it, clicks on it, or buys a product or service through it.

Unfortunately, given the competitive nature of the advertising business and the limited supply of ads, app developers commonly embed multiple ad libraries into their mobile apps. These multiple ad libraries would then request ads from various ad-pools that are offered by different advertising companies. More precisely, an app would request an ad using one library. If the library cannot *fill* the ad, the app would request an ad from another library. The average *fill rate* for ads is as low as 18% [3]. Such a low fill rate encourages developers to use multiple ad libraries to maximize their chances of obtaining an ad to display to their users.

2.2 Related Work

Prior wok primarily investigates privacy and security issues of ad libraries [4, 7, 9]. In contrast our work explores software engineering aspects of ad libraries, namely updates such libraries.

Grace *et al.* analyzed 100 libraries, and found that most ad and analytics libraries collect personal information, and some of them allow the possibility to execute code unauthorized[9]. Enck *et al.* analyzed the top 1,100 popular apps from Google Play. Their analysis reveals that ad and analytics libraries are not dangerous per se, but often exploit, and insecurely transmit, tracking identifiers to their servers [7]. Davidson and Livshits conducted an experiment on 3,120 Android apps from which they were able to eliminate 2,684 unnecessary permissions. They found that the most common permissions are INTERNET (699 apps) and ACCESS_COARSE_LOCATION (502 apps) [4]. Other researchers have even looked into the power consumed by the ads in apps. For example, Pathak *et al.* find that free apps can drain 75% more power due to poorly developed ad libraries [15].

3 What are the expenses related to ad library updates?

This section discusses the major expenses involved with ad library maintenance. The list of expenses discussed in this section are by no means a

complete list. Rather, they comprise an initial list (*i.e.,* lower bound of expenses).

A. Software Maintenance Cost: One of the obvious cost factors of ad library updates is that of software maintenance. The app developer needs to keep track of the updates done to the ad libraries by its developers and be on the look-out to include new ad libraries that may provide additional revenue, while remembering to remove ad libraries that violate any new app store policy. Once a new version is available, app developers need to embed the new version of the ad library and update the app code as well to reflect the new version of the library, i.e., replace calls to the old ad API by calls to the new one. Integrating a new version of an ad library might even entail re-designing part of the UI to fit in the new ad library. This cost is applicable for new ad libraries that are added and old ones that are removed. Once changes to the code are done, the app now needs to be re-built and re-tested. All of these activities take time and effort [12], thereby adding to the usual software maintenance cost for the app.

B. Delays in Delivering the New Version of the App to the End User: Once an app has been updated by the developer for maintenance related ad library updates, it has to be deployed in the app store. Even in app stores like Google Play where the deployment system is fully automated, it can take a few hours before the new app is available in the app store. However, in stores like the Apple App Store, this deployment phase can take days or weeks, since each version is manually verified. Once the new version of the app is available in the app store, there still remains the uncertainty as to when the end users would update the app in their mobile devices. New releases are not necessarily welcomed by users, since there is always a risk that a new update (even if it only involves just ad library updates) breaks an app's behaviour [11]. All of these delays would result in lost revenue.

C. Lack of Ads: App developers could potentially lose revenue when they do not update ad libraries. Such a loss of revenue is due to a lack of ads being served from dead ad libraries [8]. We define *dead ad libraries* as ad libraries that remain in the app without being able to serve any more ads. The developers could also lose revenue when the ads delivered through an old library are not as interesting from a user perspective as ads delivered through a new library. This is because if the users are not interested, then they may not click on an ad in the app, which is one of the major ways that generates revenue for app developers.

D. Poorer End User Experience: Advertising companies also update

ad libraries to provide better ads, to improve performance, and to fix bugs. When developers ignore these changes, end users can be left with ads that are not relevant to them, consume too much CPU or battery life, or in the worst case access irrelevant private information. Across all app stores, we can find end user complaints about ads within apps [14]. These comments are often associated with poor ratings, and thus affect future downloads of an app. Furthermore, the users often state that they will either not use the app, or worse, uninstall it. In the former case, app developers will not be able to get ad revenue from these users, and in the latter case, even if the app is updated to address the ad-related issues, the end user will not get the notification for an update.

Key Take Away: Additional maintenance activities, delays in delivery, lack of ads, and poor end user experience can directly or indirectly impact the app developers financially.

Hence, it is important to understand how prevalent ad library updates are for mobile apps. However, no concrete numbers or studies are available for this topic [17]. Existing studies and surveys only consider the cost of developing the first version of an app, with only rough approximations for maintenance cost [1, 16]. In this paper, we present an empirical case study to quantify the prevalence of ad library updates in mobile apps.

4 Identification of Ad Libraries

To investigate the prevalence of ad library updates, we first need to identify ad libraries. Since there exists no collection of existing ad libraries, we identify them from data obtained via a crawl of the Google Play Store [6]. The crawl downloaded all free-to-download apps in the store throughout 2011, and resulted in 120,981 Android apps with 236,245 versions in total. Using this dataset, we can identify ad libraries and their distribution throughout a large number of apps.

The crawled data contains the Android Packages (APK) of each app (not their source code). We first extract the Java bytecode from the APKs using dex2jar [1], then use the *Apache bcel* library [2] to extract for each class in each app, the fully qualified class name (package or namespace in which a class is contained and the class name) and its set of method names. We then

[1] http://code.google.com/p/dex2jar

[2] http://commons.apache.org/bcel/

manually filter the fully qualified class names of all apps with the regular expression [aA][dD] (*e.g.*, *com.package**Ad**libraryName.**Ad**className*).

The very basic regular expression (*i.e.,* [aA][dD]), leads to a large number of matched class names, even though these matched classes would not correspond to ad libraries. Hence, we needed to manually validate the matched classes. For this, we grouped and sorted the fully qualified class names according to their frequency (*com.google.ads.AdActivity* with 149,321 occurrences was the most popular class). For each fully qualified class name that had a frequency of more than 200 (*i.e.,* included in at least 200 apps or 0.1% of the total number of apps in the crawled data), we did a web search of the package name in order to find the website for the advertising company of that library. We expect that each advertising company has a website for developers to conclude contracts in order to receive ads and payments. In the end, we discovered 72 known ad libraries.

5 Prevalence of Ad Library Updates among Mobile Apps

We define *ad updates* as the number of ad libraries added, deleted or modified in an app. We use this as a measure of ad library updates that occurred in an app between versions.

As we do not have access to the source code of the studied apps, we resort to analyzing extracted signatures from the bytecode of these apps. Furthermore, we focus on updates in the actual ad library, but not in the glue code used to interface with the ad library, since that would require us to (1) design static analyses specific to every ad library, and (2) deal with obfuscated byte code (since we would need access to the code inside each class). Similarly, in the actual ad library, we can only measure the number of libraries that had an update, and not the amount of churn (in terms of lines of code) that happened in each library.

To measure ad updates, we only consider apps with at least two versions. As apps with very few raters could potentially be spam apps [13] and introduce a bias in our results, we also just consider apps that have at least ten raters in Google Play. This filtering leaves us with 5,937 apps with 13,983 versions in total.

We then proceed to compare how each app has updated over time. In

particular, we sought to determine if a new version of an app had ad updates, non-ad (i.e., core of the app) updates or both. In order to perform such an analysis for a large number of classes, we make use of the software bertillonage approach [5], with which we create a unique signature for each class of each app. Such signatures can be compared very efficiently. The signature is computed as follows:

1) Group fully qualified class names with the list of method names and parameters for each class into a string S. This string is the signature of the class.

2) Classify each signature S, as a specific ad library (based on the previously identified ad libraries), or classify as non-ad library code (henceforth called "core code").

3) For each such signature S, generate a hash value for quick matching. We apply SHA1 to generate the hash values.

For each app, we obtain a set of signatures for the ad library code and for the core code. We then compare the signatures of each ad library in a particular version of an app ($version_i$) with those of its subsequent version ($version_{i+1}$). Using this approach, we distinguish between the following four cases for each two consecutive versions of an app:

(a) The signatures of the ad library are identical in both versions, i.e., the APIs of this ad library did not change.

(b) Some of the signatures belong to a new ad library in app version $i + 1$, i.e., none of the class signatures in that library exist in version i.

(c) The signatures of an ad library in app version i are deleted (removed) in the version $i + 1$.

(d) The signatures of an ad library are updated in app version $i + 1$. By update, we mean that the lists of signatures of a particular ad library in app $version_i$ and app $version_{i+1}$ are different (even if just one class signature differs), and neither completely new nor completely removed, unlike the previous two cases.

Table 1: Number of app versions for each type of ad library update among the 6,850 app versions that have an ad library update.

Type of ad update	**Number of app versions**
Updated	4,470 (65.25%)
Added	2,993 (43.69%)
Removed	1,894 (27.64%)

We define that an ad library has a type of ad update if any of the (b), (c) or (d) cases occurs between versions i and $i + 1$ of an app. While the comparison of class signatures based on the software bertillonage approach is coarse grained (we will not be able to track changes made within the implementation of a particular method), the results that we obtain on the updates of ad libraries is a lower bound. Thus, the ad libraries at a minimum undergo the number of updates that we identify. If we could examine the code within a method in a class, then we would be able to identify even more updates in the ad libraries.

Finding: App developers actively maintain ad libraries: almost half (48.98%) or 6,850 out of the 13,983 versions have a type of ad library update. Table 1 presents a breakdown of the different types of ad library updates. Almost two thirds (65.25%) of the versions had at least one ad library update, while almost half (43.69%) of the versions had at least one new ad library added. However, around a quarter (27.64%) of the versions had at least one ad library removed. From Table 1, we can conclude that app developers are more actively updating their current ad libraries than adding or removing ad libraries – highlighting the importance and complexity of keeping ad libraries up to date.

Another interesting finding is that in 13.75% (942) of the app versions with ad updates, the APIs in the core code were never modified (the class signatures unrelated to the ad library classes were identical). For example, *View.CalCollection.SangGeon.Cauly* (a calculator app) was updated seven times throughout 2011, yet none of these updates modified the API of its core code - all changes were related to the *adlantis* ad library. This app is a good example of how a very stable app, like a calculator, might require constant maintenance due to updates in an ad library.

Key Finding: The prevalence of updating ad libraries in mobile apps is considerable.

6 Discussion

We now seek to dig deeper into our results to better understand why ad libraries are updated so frequently.

One of the main reasons why app developers needed to update the ad libraries so frequently in their apps was because the APIs of the ad libraries themselves were being updated very frequently. Table 2 presents how many times the APIs of the top six most popular ad libraries in the studied apps were updated, and a summary of the rationale for such updates (based on information from either the changelog files included in the ad library or from the website of the ad library). In short, some of the major reasons are: to enhance interaction capabilities between ads and app users, integrate new types of ads (e.g., video ads), fix memory management bugs, add better and more secure management of personal information, and fix bugs that prevented ads from being displayed properly.

Table 2: Type of updates performed on the most popular ad libraries

Library Name	**Number of updates in 2011**	**Brief description of the rationale for the 2011 updates**
Googleads & Admob	6	Transition for merging the *googleads* and the *admob* libraries. Bug fixes to improve the users' interaction with the served ads, and to handle clicks.
Millennialmedia	9	Ads in video format received several improvements across different updates of this ad library. Improvements to click events for ad-videos.
Flurry	8	First released for a new ad-product called App Circle. Update to pass conditions of user experience imposed by Google. Improving the checks for the availability of ads.
Adwhirl	7	Updates to work properly with different ad-networks.
Mobclix	12	Capability of interstitial ads (ads displayed before the app content is loaded). Prevent autoplay ads. Adding mediation (coordinating ad networks) capability for Google/AdMob SDK and Millennial Media Android SDK. Fixing bugs related to memory management, and clicks. Personal information is now encrypted before transmitting.
Youmi	15	Fixing bugs that prevent the display of ads on Android OS 2.3. Lightweight initialization process to display ads.

Key Finding: Ad libraries are updated by the advertising companies sometimes as frequently as 6–15 times in a year in response to financial mergers of ad companies, improved ads, and bug fixes. Such frequent app updates force app developers to update their apps frequently as well in order to include the latest version of an ad library.

7 Conclusion

We complement prior research by examining the amount of updates related to ad libraries in mobile apps. Our results show that ad library updates in mobile apps are very prevalent. Due to the various expenses that are related to ad library updates (Section 3), the maintenance of ad libraries is rapidly becoming a non-trivial and a costly affair for developers of mobile apps.

Practitioner Takeaways: Mobile app developers should be careful not to ignore ad library maintenance, since their revenues can be affected, both in a positive and/or negative way. New versions of ad libraries provide opportunities for additional income, but do require effort to integrate the new ad version in a mobile app. Choosing an advertising company that has a stable ad library with minimal updates would reduce the costs involved in updating the ad library. When app developers decide to add a new ad library to the app, they need to perform a cost-revenue analysis.

Future Work: The software engineering community has a new problem to solve: *'How to reduce the expenses and impact of ad library updates in mobile apps?'*. More specifically, since the ad libraries do not contribute to the functionality of apps, the maintenance of such libraries brings different challenges compared to traditional libraries, such as: *'How to change the API of an ad library without inducing a loss in revenue for the app developer?'*, *'How to reduce the effort involved in making these updates without any functional or non-functional field failures?'*, and *'How to push the updates to the user without disturbing them?'* These identified challenges call for future research, to improve the maintenance and quality assurance of ad supported mobile apps. Additionally, there are no studies (only informal discussions in Q & A forums or non-peer reviewed articles) to estimate the actual cost of each of the items discussed in Section 3. A large scale developer survey is needed to have a better understanding of the exact dollar values for the aforementioned expenses.

References

[1] AnyPresence Inc. The state of enterprise mobile readiness 2013. http://www.anypresence.com/Mobile_Readiness_Report_2013.php, 2013.

[2] App Annie and IDC. Mobile app advertising and monetization trends 2012-2017: The economics of free. http://blog.appannie.com/app-annie-idc-mobile-app-advertising-and-monetization-trends-2012-2017/, March 2014.

[3] H. Candeias. Smaato releases Q3 2011 mobile metrics report. http://www.smaato.com/metricsq32011/, Nov. 2011.

[4] D. Davidson and B. Livshits. Morepriv: Mobile os support for application personalization and privacy, 2012. Technical Report MSR-TR-2012-50, available at `http://research.microsoft.com/pubs/163596/MSR-TR.pdf`.

[5] J. Davies, D. M. German, M. W. Godfrey, and A. Hindle. Software bertillonage: finding the provenance of an entity. In *Proceedings of the 8th Working Conference on Mining Software Repositories*, MSR '11, pages 183–192, New York, NY, USA, 2011. ACM.

[6] S. Dienst and T. Berger. Static analysis of app dependencies in android bytecode, 2012. Tech. Note, available at `http://www.informatik.uni-leipzig.de/~berger/tr/2012-dienst.pdf`.

[7] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri. A study of android application security. In *Proceedings of the 20th USENIX conference on Security*, SEC'11, pages 21–21, Berkeley, CA, USA, 2011. USENIX Association.

[8] Google. Deprecated. Google Mobile Ads SDK v6.4.1 or lower. https://developers.google.com/mobile-ads-sdk/docs/admob/fundamentals#android, Last Visited April 2014.

[9] M. C. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, WISEC '12, pages 101–112, New York, NY, USA, 2012. ACM.

[10] Janessa Rivera and Rob van der Meulen. Gartner says mobile app stores will see annual downloads reach 102 billion in 2013. http://www.gartner.com/newsroom/id/2592315, September 2013.

[11] H. Khalid, E. Shihab, M. Nagappan, and A. Hassan. What do mobile app users complain about? a study on free ios apps. *Software, IEEE*, 2014.

[12] Kim-Mai Cutler. How do top android developers qa test their apps? http://techcrunch.com/2012/06/02/android-qa-testing-quality-assurance/, June 2012.

[13] I. Mojica, B. Adams, M. Nagappan, S. Dienst, T. Berger, and A. Hassan. A large-scale empirical study on software reuse in mobile apps. *Software, IEEE*, 31(2):78–86, Mar 2014.

[14] I. J. Mojica. Large-scale empirical studies of mobile apps. *MSc Thesis, School of Computing, Queen's University, Kingston, Ontario, Canada*, 2013.

[15] A. Pathak, Y. C. Hu, and M. Zhang. Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof. In *Proceedings of the 7th ACM European Conference on Computer Systems*, EuroSys '12, pages 29–42, 2012.

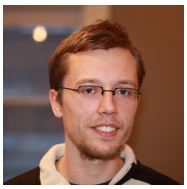
[16] Roy Chomko. The real cost of developing an app. http://www.manufacturing.net/articles/2012/07/the-real-cost-of-developing-an-app, July 2012.

[17] Username:chockenberry and Username:typeoneerror. Stackoverflow (1010 votes): How much does it cost to develop an iphone application? http://stackoverflow.com/questions/209170/how-much-does-it-cost-to-develop-an-iphone-application, Oct 2010.

[18] R. van der Meulen and J. Rivera. Gartner says mobile advertising spending will reach $18 billion in 2014. http://www.gartner.com/newsroom/id/2653121, Jan. 2014.

Israel Mojica is a software engineer at McAfee. His research interests include mobile software and empirical software analysis in general. Mojica received an MS in computer science from Queens University, Canada. Contact him at Israel_Mojica@McAfee.com.

Meiyappan Nagappan is a postdoctoral fellow in the Software Analysis and Intelligence Lab (SAIL) at Queens University, Canada. His research interests include deriving solutions that encompass all the various stakeholders of software systems and using large-scale software engineering data to also address the concerns of soft- ware operators, build engineers, and project managers. Nagappan received a PhD in computer science from North Carolina State University. He received a best paper award at the International Working Conference on Mining Software Repositories (MSR 12). Contact him at mei@cs.queensu.ca.



Bram Adams is an assistant professor at the Ecole Polytechnique de Montreal, where he heads the MCIS (Maintenance, Construction, and Intelligence of Software) lab. His research interests include software release engineering, software integration, software build systems, software modularity, and software maintenance. Adams received a PhD in computer science engineering from Ghent University. He was an organizer of the First International Workshop on Release Engineering (RELENG 13) and is a member of IEEE. Contact him at bram.adams@polymtl.ca.

Thorsten Berger is a postdoctoral fellow in the Generative Software Develop- ment Lab at the University of Waterloo, Canada. His research interests include model-driven development, variability modeling for software product lines and software ecosystems, variability-aware static analyses of source code, and mining software repositories. Berger received a PhD (Dr. rer. nat.) in computer science from the University of Leipzig. Contact him at tberger@gsd.uwaterloo.ca.

Steffen Dienst is a PhD student in the Chair of Business Information Systems at the University of Leipzig, Germany. His main research topic is using machine- learning techniques to help monitor the operation of renewable power plants. Other interests range from reverse engineering to functional programming. Dienst received a M.Sc. (Dipl.-Inf.) in computer science from the University of Leipzig. Contact him at sdienst@informatik.uni-leipzig.de

Ahmed E. Hassan is the NSERC/Black- Berry Software Engineering Chair at the School of Computing at Queens University, Canada. His research interests include mining software repositories, empirical software engineering, load testing, and log mining. Hassan received a PhD in computer science from the University of Waterloo. He spearheaded the creation of the Mining Software Repositories (MSR) conference and its research community. Hassan also serves on the editorial boards of IEEE Trans- actions on Software Engineering, Springer Journal of Empirical Software Engineering, and Springer Journal of Computing. Contact him at ahmed@cs.queensu.ca.