Back to the App: The Costs of Mobile Application Interruptions

Luis A. Leiva[1,*] **Matthias Böhmer**[2] **Sven Gehring**[2] **Antonio Krüger**[2]
[1]ITI/DSIC, Universitat Politècnica de València (Spain) [2]DFKI GmbH, Saarbrücken (Germany)
llt@acm.org, {matthias.boehmer, sven.gehring, krueger}@dfki.de

ABSTRACT
Smartphone users might be interrupted while interacting with an application, either by intended or unintended circumstances. In this paper, we report on a large-scale observational study that investigated mobile application interruptions in two scenarios: (1) intended back and forth switching between applications and (2) unintended interruptions caused by incoming phone calls. Our findings reveal that these interruptions rarely happen (at most 10% of the daily application usage), but when they do, they may introduce a significant overhead (can delay completion of a task by up to 4 times). We conclude with a discussion of the results, their limitations, and a series of implications for the design of mobile phones.

Author Keywords
Interruptions; Application Switching; Task Interleaving; Task Deferral; Resumption Lags; Large-scale Study

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI)

INTRODUCTION
Nowadays, smartphones have emerged as multipurpose devices. Besides calls and text messages, smartphones offer the possibility of installing multiple applications for a variety of purposes, e.g., gaming, browsing, listening to music, editing pictures, etc. Hence, the role of smartphones has changed from pure communication appliances to multifunctional toolsets.

So far, the amount of functionalities that are supported by mobile applications is steadily increasing. At the same time, due to the limited capabilities of smartphones and the security restrictions imposed by the manufacturers, current operating systems for smartphones lack possibilities for interplay between individual apps. As a result, instead of using a separate task-dedicated appliance, users have to switch between applications; and more elementally, applications are interrupted when unintended events like phone calls occur.

[*]Work done while visiting the DFKI institute.

From the perspective of interaction, being mobile is cognitively costly [5]. Although task interruption is an inherent problem in smartphones, little is known about the phenomenon at the fine-grained level of application usage. Yet it remains an open question how often such interruptions appear, what impact they have on the user's performance and, especially, what costs they impose on task completion times.

Researchers have investigated task interruption in the desktop domain (see next section). In this paper, we extend findings of previous in-lab studies to the domain of smartphones. We conducted a large scale observational study "in the wild" that investigated mobile application interruptions in two scenarios: (1) intended back and forth switching between applications, and (2) unintended interruptions caused by incoming phone calls. Concretely, we looked into the cost of these interruptions on task completion time. Therefore, in the context of this work, and acknowledging its limitations, we consider task completion as the time spent using an application.

We present a framework that can easily assist researchers to detect application interruptions on mobile phones at a large scale. Our findings show that interruptions caused either by incoming phone calls or by application switching rarely happen on smartphones but, as soon as they do, they might cause a significant impact on the interrupted activity. We conclude with a discussion on preventive and curative strategies, and their implications for the design of mobile phones.

RELATED WORK
Task interruptions have been extensively studied on stationary PCs (see, e.g., [1, 3, 4, 7, 8]), but little has been reported for mobile users and their unique set of difficulties. It is clear that mobility imposes cognitive restrictions and continuous interruptions on application usage; however, to the best of our knowledge, there is no previous research that focuses *exclusively* on the application level. Moreover, other studies in a similar vein [5, 6] have been performed in carefully controlled settings. Our main aim is to investigate the costs of mobile application interruptions "in the wild", i.e., in a natural, general environment *and* at scale.

Oulasvirta et al. [6] carried out a study (28 subjects) on mobile Web search tasks while moving, and observed that continuous attention to the mobile device is fragmented, mostly due to environmental distraction, and broke down into short time spans. Karlson et al. [5] focused on the tasks as a whole, including switching to a PC, if necessary, to complete them. They carried out a 2-week diary study mostly focused on email management (24 subjects), characterizing how problematic interruptions are to mobile users and identifying primary sources of frustration. In addition, Böhmer et al. [2]

looked into unconstrained mobile application usage, reporting descriptive statistics on what is probably one of the largest datasets in mobile HCI today. However, they did not study how often interruptions happen and what their real impact is on the user's productivity.

STUDY

We analyzed the *AppSensor* dataset provided by Böhmer et al. [2]. *AppSensor* is an Android background service that indicates the currently used (foreground) application at a sampling rate of 2 Hz. It also provides data related to phone usage such as runtimes or timestamps of screen on/standby. *AppSensor* collapses measured values into single data samples, with one sample denoting which application was launched at what point in time and for how long it was used; e.g., the phone application was launched at 6:30 PM and then was used for 32 minutes. A detailed description of this corpus as well as the *AppSensor* tool can be found in [2].

The dataset contained around 5.5M data samples, corresponding to 15.7K different applications used for one year and a half (from August 2010 to January 2012, Table 1) by 3.6K unique users who were geographically distributed worldwide.

Data samples	Days of study	Applications	Users
5,495,815	532	15,756	3,611

Table 1: Features of the analyzed dataset.

Method

Detecting Interruptions

As observed in Figure 1, an interruption is considered when the foreground activity changes from application x to y ($x \neq y$) and then returns to x. Additionally, as indicated in the figure, we impose the following restrictions: *1)* Launcher/dialer applications (L) and calls (C) are not considered to be interrupted. *2)* An application cannot be interrupted by L. More formally, $L \neq x \neq C$ and $y \neq L$. This way, applications can be deferred (i.e., *interrupted on purpose*) or interrupted by an *incoming* phone call. For the sake of brevity, we will refer to the former case as **internal** interruptions, and will use **external** for the latter.

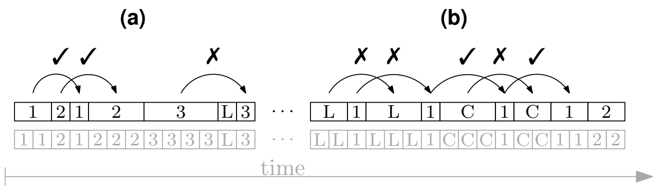

Figure 1: Light-gray boxes illustrate app identifiers from the data samples, which were used to recompose the sequences of each user's activity (black boxes) and detect **internal** [1a] and **external** [1b] interruption patterns.

It is worth noting that applications can be deferred for a long time because of environmental distractions not related to mobile phone usage (e.g., being prompted by a friend to chat for a while), in which case the device switches to stand-by mode. Because of this, to avoid misleading results, we use a time window of 1 minute; so that if a series of consecutive data samples were found for the same user + day + app, such application was considered as a different activity if the time between data samples exceeded 1 min.[1] We chose this specific duration for the time window based on the average application usage length according to the literature [2, 10].

Measuring Interruptions

Let T_n be the runtime of an application in *normal* operating condition (i.e., without interruptions) and let $T_r = T_b + T_a$ be the runtime of the same application when it is interrupted, which is decomposed into the runtime *before* the interruption T_b and the runtime *after* the interruption T_a (until the application is closed, or another interruption is detected). As shown in Figure 2, $T_o = T_r - T_n$ is the overhead imposed on the application as a consequence of the interruption T_i. Overheads are also cited as "resumption lags" in the literature [3, 8], and usually lead to a decrease in primary-task performance. However, notice that, while $\{T_a, T_b, T_i, T_n\} \in \mathbb{R}_0^+$, $T_o \in \mathbb{R}$, i.e., overheads can be either positive or negative, since an application that is used in normal conditions for a very long time might not be resumed when the user recovers from an interruption, yielding $T_r < T_n$ and hence $T_o < 0$.

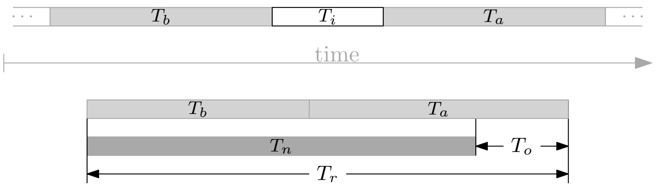

Figure 2: Computing an application overhead when interrupted.

Notice also that overheads can only be fairly computed for paired cases, i.e., one needs to compare the time an application is interrupted against its normal usage time for a given user. Nonetheless, while mining the dataset we computed all possible cases (including unpaired conditions), in order to quantify precisely how often interruptions did happen (Table 2). Then, unpaired cases were filtered and dropped from the subsequent analysis (Table 3 and Results section).

Procedure

Logs were sequentially grouped per day per user. Then we processed all interaction sessions, which are summarized in Table 2, and computed a series of descriptive statistics, depicted in Table 3. These values were macro-averaged, to give equal weight to each user and their applications. Outliers were considered when the mean exceeded 1.5 the interquantile range. To highlight the differences between both types of interruptions, we carried out different hypothesis tests. Interaction effects were considered at the $p < .05$ level.

RESULTS

As previously mentioned, we report here the results of our study for **external** vs. **internal** interruptions after removing

[1] Concretely when $t_j - (t_{j-1} + r_{j-1}) > 1$min, where t_j is the jth log timestamp and r_{j-1} is the application runtime.

	external	internal
Interruption data samples	776,922	970,543
Interrupted users	1,929 [1,676]	2,926 [2,609]
Interrupted applications	1,373 [487]	4,626 [1,043]

Table 2: Dataset summary after processing the logs, showing in brackets the number of balanced (paired) cases.

outliers and unpaired cases (final sample sizes are denoted in brackets in Table 2).

	external	internal
Daily interruptions (% usage)	3.2 (2.2)	8.3 (5.3)
Interrupted applications	3.3 (2.6)	8.7 (7.2)
Regular app. runtime (s)	24.8 (31.8)	18.9 (24.4)
Runtime when interrupted (s)	107.1 (121.1)	87.9 (75.5)
Interruption duration (s)	12.5 (8.1)	23.7 (19.3)
Overhead duration (s)	43.2 (65.9)	34.4 (40.7)

Table 3: Mean (and SD) values per user (first 2-rows) and per app.

Since data could not be considered as normally distributed[2], we used the Kolmogorov-Smirnov test, which is non-parametric[3] and hence does not make assumptions about data distribution.

Unsurprisingly, **internal** interruptions are more frequent ($D^+ = 0.50, p < .0001$, Cohen's $d = 1.24$) and involve more applications ($D^+ = 0.36, p < .0001, d = 0.98$) in comparison to when users are interrupted by phone calls, as shown in the first 2 rows of Table 3. These differences were found to be statistically significant.

Overall, applications interrupted by phone calls take more time to complete than switched applications ($D^+ = 0.06, p = .04, d = 0.18$). Notice the differences on application runtime when compared to its normal usage ($D = 0.09, p = .003, d = 0.20$). The duration of phone calls was found to be significantly smaller than the duration of the switched applications ($D^- = 0.31, p < .0001, d = 0.75$), however phone calls produce a significantly higher overhead time on the interrupted application compared to the **internal** pattern ($D^- = 0.08, p = .006, d = 0.17$). These results suggest that people usually engage more when using applications, and that **external** interruptions are more disruptive regarding task recall.

We computed the correlations between these measurements, and they were found to be mostly weak/moderate (Figure 3). The most consistent correlations found both in the **external** and **internal** groups were, as expected, those of *1)* the runtime of an interrupted application vs. its overhead; and *2)* number of daily interruptions vs. number of interrupted applications.

In sum, it was interesting to observe that, overall, mobile interruptions at the application level do not happen often: around 3% of daily usage in case of phone calls, 8% when

[2]Verified by previous Shapiro-Wilk tests ($p < .0001$ in all cases).
[3]Kolmogorov's D statistic refers to two-tailed comparisons, while D^+ and D^- refer to one-tailed comparisons.

	n_i	n_a
n_i	–	0.26***
n_a	0.27***	–

(a) Per user.

	T_r	T_i	T_o
T_r	–	0.12**	0.68***
T_i	0.09*	–	0.13**
T_o	0.51***	0.22***	–

(b) Per user applications.

Figure 3: Correlation study. Above main diagonal: ρ for **external** interruptions. Below main diagonal: ρ for **internal** interruptions. [3a] n_i: number of daily interruptions, n_a: number of interrupted applications. [3b] T_r: interrupted application runtime, T_i: interruption time, T_o: overhead time. Statistical significance: $^*p < .05$, $^{**}p < .01$, $^{***}p < .001$.

switching back and forth between applications; but, when they do, the resumption cost may be exceedingly high.

DISCUSSION

The extent to which an interruption occurs can be a helpful means for designing smartphone interaction. Also, knowing both the duration of the interruption and the resulting overhead can help to improve the design of applications that are aimed to support multitasking to a greater or a lesser degree.

An obvious approach to reach these goals is helping the user regain the context of the deferred application when it is resumed. On the desktop arena, when this happens, pertinent visual cues are given as a help for easing the recovery from the interruption; however, this is usually not applied to smartphones [5]. Iqbal and Horvitz [3] offered two directions for recovering from interruptions on the desktop: reminding users of unfinished tasks and assisting them in efficiently recalling task context. We believe that, in the mobile domain, the key factor is reducing the overhead time that is introduced by an application interruption. To do so, we suggest either helping the user to maintain the context while switching to another application, or to support regaining context when returning to the interrupted application. Next, we describe design considerations that can support these recommendations.

Design Implications

Our results entail a series of interventions for designing mobile interaction to reduce the overhead that is introduced by application interruptions. In general, inspired by previous approaches in the field of desktop interruptions, we distinguish between *preventive* (preparing the user for being interrupted, cf. [9]) and *curative* (supporting the user after being interrupted, cf. [3]) strategies.

Preventive: Preparation for Being Interrupted

When a task interruption occurs, the user could be prepared to leave the current task. For instance, for incoming phone calls the caller usually waits on the line for some seconds. Postponing the call a bit more (say, 500 ms) might provide time to give the user an auditory/visual/haptic signal that soon the phone application will pop-up. This way, the user would be able to save a mental state and keep in mind the recently interrupted application before he is interrupted. Further, currently users receive a full-screen visual notification of incoming calls on most smartphones. We believe that gradually overlaying this notification onto the currently used application would also provide the user with the possibility to take a subconscious snapshot of his most recent action. This particular approach may apply equally to the **internal** interruptions.

Curative: Guidance for Going Back into Tasks
When the user resumes a previously interrupted application, she has to reallocate cognitive resources, which becomes increasingly difficult if the resource demands were high to begin with [3]. Therefore, she should be given some help to be able to immediately (and easily) continue with it. For example, this could be achieved by automatically replaying the last N milliseconds of UI interactions, to give a hint of what she was doing before the interruption. The system could also leave a visual on-screen cue such that the user could remember at any time to which task she is switching back. Alternatively, when returning to the interrupted application, the canvas of the foreground application could vanish into the direction of the last focus of interaction, in order to guide the user to the screen position before the interruption took place.

Limitations of the Study

First, our results are dependent on the quality of the dataset. As previously pointed out, there exist many disruption contenders apart from mobile application usage itself that lead people to change tasks and applications [1]. In addition, while the overhead measure has been used in the literature for routine tasks, it is ideally suited to resumption where there is little re-encoding of the primary task required (see [7]). These facts and other environmental factors are an important source of indeterminism that we were unfortunately not able to measure within the data. Although we can report that interruptions *do* happen at the application level and that their cost is significant, our observational study was non-controlled, based on pure data mining. Hence, a line of future work will be enriching the data collection process with additional experience sampling approaches. This would provide us with deeper insights about tasks being interrupted.

Furthermore, we acknowledge that a direct mapping between tasks and applications is hard to convey, as a task can involve a single application (e.g., reading a document) or many (e.g., to prepare a meeting one can usually check the calendar, consult a web page, and take some notes). As such, it is not clear to what extent the ecological impact of mobile application interruptions would align with user's cognitive load and higher-level goals; e.g., one cannot guarantee if a 2-minute phone call would really significantly interrupt a user preparing a meeting.

Finally, another observation associated with this analysis is that, besides having used a large population with different backgrounds in a realistic context, the analyzed user sample comes from one of the mainstream mobile ecosystems. As such, further research would be needed to understand application interruptions in other platforms and verify whether our findings still hold.

CONCLUSION

In this paper, we replicated findings of previous lab studies on desktop interruptions and extended them to the mobile domain, by exploiting a large-scale dataset of mobile application usage. We have observed that app-switching behavior as well as incoming phone calls are a non-negligible source of disruption and therefore should be mitigated. Our study reveals three general findings:

1. Application interruptions rarely happen on smartphones, but when they do, they can be really costly for the user. This poses a wealth of new challenges for mobile designers and smartphone vendors.
2. App-switching behavior does not happen as often as it is presumed. While smartphones allow changing the focus of interaction, users are reluctant to do so. One reason might be that there are no mechanisms or suitable interaction techniques (yet) to support regaining context after switching between mobile applications.
3. Phone call interruptions add a significantly high overhead on the interrupted application in comparison to those of app-switching. This was expected, as incoming calls potentially can happen anytime. However, it was surprising to note the cost on the interrupted application: the runtime could be increased by up to four times.

Furthermore, we have discussed possible approaches to reduce the overhead caused by application interruptions and to help users resume task flow. Aside from the limitations of the study, it is our belief that our observations can lead to helpful guidelines for mobile interaction design.

ACKNOWLEDGMENTS

We thank Duncan Brumby and our reviewers for their valuable suggestions to strengthen this paper. We also thank Antti Oulasvirta for providing initial feedback. This work is partially supported by the MIPRCV project (CSD2007-00018).

REFERENCES

1. Benbunan-Fich, R., Adler, R. F., and Mavlanova, T. Measuring multitasking behavior with activity-based metrics. *ACM TOCHI 18*, 2 (2011), 7:1–7:22.
2. Böhmer, M., Hecht, B., Schöning, J., Krüger, A., and Bauer, G. Falling asleep with Angry Birds, Facebook and Kindle – A large scale study on mobile application usage. In *Proc. MobileHCI* (2011), 47–56.
3. Iqbal, S. T., and Horvitz, E. Disruption and recovery of computing tasks: field study, analysis, and directions. In *Proc. CHI* (2007), 677–686.
4. Jin, J., and Dabbish, L. A. Self-interruption on the computer: a typology of discretionary task interleaving. In *Proc. CHI* (2009), 1799–1808.
5. Karlson, A. K., Iqbal, S. T., Meyers, B., Ramos, G., Lee, K., and Tang, J. C. Mobile taskflow in context: A screenshot study of smartphone usage. In *Proc. CHI* (2010), 2009–2018.
6. Oulasvirta, A., Tamminen, S., Roto, V., and Kuorelahti, J. Interaction in 4-second bursts: the fragmented nature of attentional resources in mobile HCI. In *Proc. CHI* (2005).
7. Salvucci, D. D. On reconstruction of task context after interruption. In *Proc. CHI* (2010), 89–92.
8. Salvucci, D. D., and Bogunovich, P. Multitasking and monotasking: The effects of mental workload on deferred task interruptions. In *Proc. CHI* (2010), 85–88.
9. Trafton, J. G., Altmann, E. M., Brock, D. P., and Mintz, F. E. Preparing to resume an interrupted task: effects of prospective goal encoding and retrospective rehearsal. *Int. J. Hum.-Comput. Stud. 58*, 5 (2003), 583–603.
10. Vaz De Melo, P. O. S., Akoglu, L., Faloutsos, C., and Loureiro, A. A. F. Surprising patterns for the call duration distribution of mobile phone users. In *Proc. ECML PKDD* (2010), 354–369.