Interrupted by a Phone Call: Exploring Designs for Lowering the Impact of Call Notifications for Smartphone Users

Matthias Böhmer[1], Christian Lander[1], Sven Gehring[1], Duncan Brumby[2], Antonio Krüger[1]
[1]DFKI GmbH, Saarbrücken, Germany; [2]UCL Interaction Centre, London, UK
[1]{matthias.boehmer,christian.lander,sven.gehring,antonio.krueger}@dfki.de; [2]brumby@cs.ucl.ac.uk

ABSTRACT
Mobile phones have evolved significantly in recent years from single-purpose communication devices to multi-purpose computing devices. Despite this evolution, the interaction model for how incoming calls are handled has barely changed. Current-generation smartphones still use abrupt full-screen notifications to alert users to incoming calls, demanding a decision to either accept or decline the call. These full-screen notifications forcibly interrupt whatever activity the user was already engaged in. This might be undesirable when the user's primary task was more important than the incoming call. This paper explores the design space for how smartphones can alert users to incoming calls. We consider designs that allow users to *postpone calls* and also to *multiplex* by way of a smaller partial-screen notification. These design alternatives were evaluated in both a small-scale controlled lab study as well as a large-scale naturalistic in-the-wild study. Results show that a *multiplex* design solution works best because it allows people to continue working on their primary task while being made aware that there is a caller on the line. The contribution of this work is an enhanced interaction design for handling phone calls, and an understanding of how people use it for handling incoming calls.

Author Keywords
Smartphones; app usage; phone calls; interruptions.

ACM Classification Keywords
H.5.m Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION
People's smartphones are used to support a large variety of activities and tasks. Indeed, some have gone so far as to suggest that the smartphone will become the primary computer of choice for many users [28]. People can use their smartphones for a variety of work and leisure activities, from processing email and managing appointments in their calendar, to listening to music and surfing the web. The smartphone is, however, still fundamentally a telephone. When a user receives a call, current-generation smartphones tend to notify the user with a full-screen visual notification. This notification abruptly forces the user to stop whatever task they were previously occupied with and attend to the call. For instance, a user might be partway through entering the time and location of a meeting to a calendar from an email. When a call is received, this activity must be suspended and returned to after deciding how to handle the call. During that time, the user might have forgotten the location of the meeting and so have to look it up again – that is, if they remember to complete the task at all.

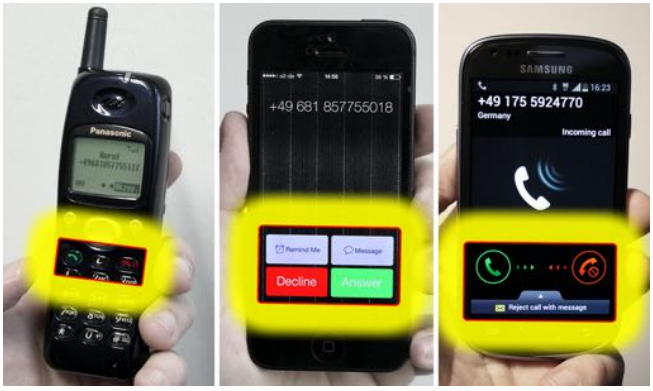


Figure 1. Only slight evolution occurred in phone call apps while mobile phones evolved from mere phones to multifunctional tools.

This vision of how people interact with their smartphones is supported by the results of a recent large-scale in-the-wild study: Leiva et al. [16] analyzed data from several thousand users over an 18-month period. They found that smartphone users are rarely interrupted by phone calls while they are using other apps (at most 10% of daily app usage). But when they are interrupted, it is massively disruptive and increases the time it takes users to complete the task they were working on prior to dealing with the call. Given the disruption caused by incoming phone calls, we consider whether there is potential for revisiting the design space for how they are handled by smartphones.

When we consider how smartphones notify users of incoming calls, it is quite clear that the basic interaction model has not changed since the development of early mobile phones. Figure 1 juxtaposes a Panasonic mobile phone from circa 1999 (on the left) beside current generation phones. It is clear that apart from the fact that hardware buttons have been replaced with touchscreen buttons, the basic interac-

CHI 2014, April 26 - May 01 2014, Toronto, ON, Canada
Copyright 2014 ACM 978-1-4503-2473-1/14/04$15.00.
http://dx.doi.org/10.1145/2556288.2557066

tion model is the same: the user is alerted to an incoming call and has to decide whether to accept it or decline it. This is despite the fact that the current generation of smartphones affords far greater functionality and support for tasks than its earlier predecessor. Current call answering screens allow declining calls with text messages, or setting up reminders (see Figure 1). While this helps in catching up with declined calls later, it still leaves the problem of full-screen alerts interrupting a user's concurrent app usage.

This paper revisits the design of mobile phone call UIs with the goal to better handle interruptions caused by incoming phone calls during app usage. We focus on cases where the user is interrupted by an incoming phone call while they are engaged in another ongoing activity on their smartphone. We make three contributions: First we extend the activities of call handling, explore the design space, and present two implementations in that space. Second, we report the results of a controlled lab study that evaluates the effectiveness of these various design alternatives for mitigating the effects of call interruptions. Finally, we describe a large-scale in-the-wild study that was conducted following the release of a call-handling app to an app store. From this study we learned how our design was used in natural contexts.

RELATED WORK

There is a long tradition in the HCI community of studying the effect that interruptions, such as handling an incoming phone call, have on task performance [9,17,20]. We review this work along with work that has sought to develop smart systems to handle calls better. Finally, we discuss what few attempts there have been to develop commercial apps to tackle this problem.

Interruptions Are Disruptive

It is well understood that interruptions disrupt ongoing activities and take time to recover from. Memory for goals [3,27] has emerged as an important theoretical framework for understanding how people re-engage with a task following an interruption. The theory assumes that people use their memory as well as salient cues in the environment to help reconstruct what it was they were doing prior to being interrupted. This process takes time, and is referred to as a *resumption lag*.

There is evidence that incoming calls incur a significant resumption lag for smartphone users. Leiva et al. [16] found that after finishing a phone call, it took users up to 40 seconds longer to finish the task they were working on prior to dealing with the call. This observation led Leiva et al. [16] to suggest that current-generation smartphones should move away from using an *immediate* full-screen notification to signal an in-coming call, because this does not give the user any time to prepare for the interruption. Instead Leiva et al. suggest that a gradual overlay notification should be used. The idea is that this would give the user time to prepare for the call. Consistent with this idea, Iqbal et al. [13] found that call pre-alerts can reduce the impact that incoming calls have on driving performance – presumably because this gives people an opportunity to think about whether or not it is a good time to take the call.

There is good evidence to suggest that given greater flexibility and choice people will choose to defer an interrupting task until after a task (or a subtask) has been completed [5,14,17,20]. For instance, Fischer et al. [8] show that most opportune moments for mobile interruptions are after episodes of device usage (e.g., after sending an SMS message). Hence, if a user is working on their smartphone, they might prefer to be given the opportunity to defer an incoming call until after they have completed the task they are working on. Some reports suggest that up to 30% of calls are missed, often for intentional reasons [19]. Hence, there is scope to consider alternative design options for handling calls that reduce the level of demand on the person receiving a call.

Making Calls Less Disruptive

There has been a large body of work that has attempted to design systems to reduce the disruption caused by incoming phone calls. There have been numerous attempts to build adaptive notification systems that manage when calls are allowed. Iqbal et al.'s *OASIS* [14] system holds non-urgent computer alerts until periods when users are interruptible. Ter Hofte [26] has applied this idea to managing telephone calls by building a predictive model that blocks calls to users when they are actively engaged in an activity. In a similar vein, Ho and Intille [12] presented a sensor-based strategy for delaying call interruptions that are not time-sensitive until a physical activity transition. Stamm et al. [23] have also developed a system that calculates the cost of interruptions on mobile phones so that they can be better scheduled. However, they argue that modeling the scheduling of phone call interruptions is not easy because of the synchronous nature of telephone communication.

The sharing of context information has been put forward as one way to overcome the issues imposed by the synchronous nature of telephone communication. *ContextCalls* was an early system proposed by Schmidt et al. [21] to mitigate the problem of call interruptions by making the callee's context transparent to the caller and vice versa. Taking a similar approach, *TellingCalls* by Granndhi et al. [10] conveys information about the call between caller and callee (e.g. the call's topic). Knittel et al. [15] have developed a system that augments the personal address book in a phone with information to help people make an informed decision about whether they should call (and maybe interrupt) the callee. Indeed, this idea has been realized in voice-over IP systems, such as Skype, which allow users to make explicit their availability. Ironically, though, Teevan and Hehmeyer [25] found that people are actually more likely, rather than less likely, to accept a call when their status is "busy" or "do not disturb". A possible explanation for this might be a self-selection bias, such that only important calls are initiated to people with a busy status. Regardless, there is still the problem that users are not very good about updating their status with such context sharing systems.

Commercial Applications

There are a few commercial apps available on the app market to help user manage calls better (e.g. *SmallCall*, *A+ Call Manager*). Simply silencing an incoming call has become a feature on many devices. However, no scientific insights have been generated from these apps. This paper explores a more comprehensive design space going beyond the isolated solutions of these apps.

In summary, little is known about the problem of task interruptions with the primary task running on the phone. This is what this paper aims to address by presenting a design for lowering the impact of call interruptions. Based on the first observations of Leiva et al. [16] the goal of this paper is to increase the understanding of phone call interruptions and to propose new UIs to reduce their effects.

REVISITING THE PHONE CALL UI

Analyzing current smartphone models (iPhone, different Android devices, Windows Phone, N9) we found that they have two shortcomings that may amplify the disruptiveness of incoming call notifications:

1. Call apps by default use full-screen modal dialogs to notify the user of incoming calls. This visually detaches the user from his previous app and thus might lead to a higher impact of the interruption.
2. Call apps only provide the user with two options: to promptly either accept or decline an incoming call. This unavoidable decision (accept vs. decline) might amplify the interruption. Further, accepting the call pulls the user's attention further away from the previous app to the phone conversation, and declining may have additional negative side effects (e.g. social implications).

We tackle these two issues by revisiting and extending the design space of phone call apps as follows. First, we increase the user's freedom in deciding when to pick up a call by introducing the possibility of postponing an incoming phone call. Second, we re-iterate on the design of user interfaces of phone call apps to mitigate the interruptive effect of incoming calls while an app is being used. In particular, we extend the design of current phone call UIs to allow for a higher degree of multitasking and additional options to handle incoming calls.

Figure 2 describes an activity diagram for handling calls from incoming (I) to ending (E). The chart highlights the new activity and transitions that we propose in this paper (green). In addition to accepting (A) and declining (D) an incoming call, we introduce postponing (P) the call. The three activities A, D, and P relate to handling of notifications of incoming calls. While calls can be accepted or declined only once, the postpone activity can be repeated several times. The incoming call (I) and a postponed call (P) might directly end (E) if the caller hangs up or the voicemail answers the call.

Current Phone Call User Interfaces

Figure 3a sketches the design of currently predominant phone apps, which we refer to as *baseline UI*. This design

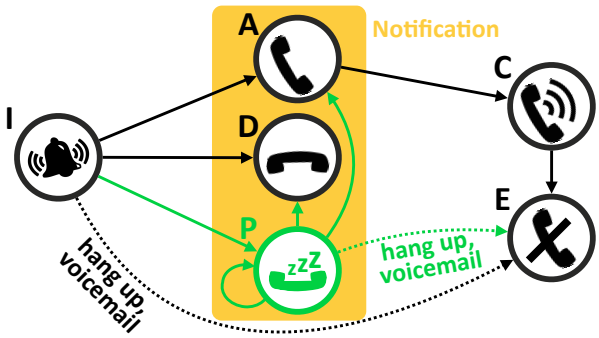


Figure 2. Activity diagram of handling incoming calls.

only provides options for either accepting or declining the incoming phone call. Some specific implementations provide additional options, e.g. shortcuts to sending messages like *"I am currently in a meeting"* to the caller when declining the call, or augmenting the call with additional information like the caller's profile picture or birthday.

This baseline UI results in overheads in usage times of the concurrent apps when interrupted [16], for instance, if a user is writing an e-mail or updating their status on a social network. As described earlier (see Figure 1), this interaction model has barely changed since the development of early mobile phones. This seems like a missed opportunity as current-generation phone operating systems allow users to multitask between different apps. Despite this potential, current-generation smartphones do not allow users to simultaneously use an app while there is an incoming call.

Postponing Incoming Calls

A first improvement to mitigate the effect of call interruptions is to give the callee a greater choice in deciding how to handle the call. Besides the option to accept or decline a call, we provide a third option: to postpone a call. Hence, the user can return to his previous app without a need to decide how to react to the incoming call. The approach of postponing calls transfers a user's ability to pick up the call at will from landline phones to smartphones. Users benefit from the increased flexibility and choice to defer the call interruption. After some time, however, voicemail might possibly answer the call. Postponing is not as determined as a "decline and call-back" strategy because the caller will not recognize the callee's postpone action.

Figure 3b shows that the postpone option can be implemented as an additional button besides accept and decline within the full-screen notification for calls. When a call is postponed, the phone call app should go into the background so that the user can continue working in his previous app. The caller will be kept waiting on the line. After a certain time span the call app will come to the foreground again, and again the user has the three options to accept, postpone or decline the call. We will refer to this proposal for a new call interaction design as the *postpone UI*.

The difference between 'letting it ring' and pressing 'postpone' is that the notification UI disappears when pressing 'postpone', and reappears automatically after some seconds.

For the caller, however, there is no difference and it will keep ringing until he hangs up, or possibly the callee picks up, or the voicemail answers.

Multiplexing Notifications

Multiplexing the primary and secondary tasks on a smartphone's limited screen real estate has been found to provide the user with more control while being interrupted [2]. Therefore, a second approach to mitigate phone call interruptions is to alter the visual appearance of call notifications. Rather than having a full-screen notification, we propose to divide the mobile screen's limited space into two areas (see Figure 3c). The basic idea is to use a smaller area of the screen to notify the user of an incoming call, rather than using a full-screen notification. And again, the user has the choice to either accept, decline, or postpone the call. With less screen area used for the alert, the user can continue working on their primary task. For example, they might want to finish writing a sentence in an email or tag a point on a map. We will refer to this design as the *multiplex UI*.

USER STUDY I: IN THE LAB

We conducted a study to test the two proposed UI designs. We were interested in how the interruption of an incoming phone call would impact user experience and app usage.

Study Design

We followed a within-subject A/B/C design for the controlled lab study. We tested the multiplex UI compared to the postpone UI and the baseline UI. In each condition there was a primary task people had to solve using some apps, and a secondary task with a phone call that interrupted the app usage of the primary task.

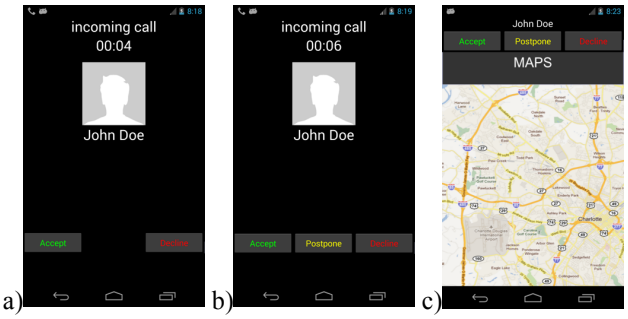
Participants

Twelve participants (six female) with mean age 25.8 years (SD = 4.1 years, range: 19 - 31) were recruited from a local university campus. Participants had a mix of technical and non-technical backgrounds.

Materials

We implemented the presented design options for call notifications as a prototype for Android phones, and we gave an instrumented smartphone to the participants. Figure 3 shows screenshots of the three call-handling UIs we implemented according to the design space introduced before. The postpone duration was set to 5 seconds.

For the primary task we implemented common mobile use cases inspired by the tasks of Cauchard et al. [6], who used typical mobile apps like maps, contact list, and calendar. Participants got a question and had to use three other apps for answering the question, always by memorizing and connecting pieces of information shown in the other apps. Within the interruptive phone call our participants had to do a word-generation task [24]: the caller would say five words, and the participant had to think of and respond with new words starting with the last letter of the given word (which was in the participant's mother tongue). As keywords the peers would say "hello" and "bye" to signal start



a) b) c)

Figure 3: Screenshots of prototype implementing the three UI designs: a) baseline UI; b) postpone UI; c) multiplex UI.

and end of the task. The call was automatically initiated on the caller's phone when the participant started working on a task and the interruption reached him after 6 seconds.

A pre-study revealed that users might intentionally postpone the call until they have solved the task, since there is no hurry to accept the simulated call. To mitigate this effect, we first introduced a random time (between 20-30 seconds) after which the caller would hang up (as if the voicemail answered the call). In addition, to motivate people to perform well in both tasks, we gave an additional award (20 EUR gift certificate) to the participant who performed best in the whole study (correctly solved tasks, correctly created words, and fastest in both tasks).

Procedure

We explained to our participants what the study was about. We spent about 15 minutes to acquaint them with the tasks, as well as the different designs: Participants learned how to use the three different call UIs and how to solve the primary tasks. They were also introduced to the secondary task on the phone, both standalone and as an interruption during the primary task. For the training we used questions and words not used in the three main parts of the study.

After the training the study had three parts: In each part the participants had to answer 20 questions in the primary task, 11 of them being interrupted by calls. During training the participants were instructed to be the one to hang up after each call. Each part was assigned to one condition (baseline UI / postpone UI / multiplex UI) in a counterbalanced way. After each part, the participants were asked to fill out a questionnaire. One experimenter stayed with the participant, while a second experimenter carried out the call interruptions remotely. At the end of the study we asked for additional demographic information. The experiment took about 60 minutes to complete and participants received a 10 EUR gift certificate for their time.

Measures

We collected logging data from the device to measure how people interacted with the UIs and the applications of the primary task, and we collected measures from questionnaires that we administered after each part. In particular, we observed the following dependent variables:

- Time on task: We logged how long people were working on the primary task (**TOT**), i.e. how long they were working with the four apps (question, agenda, map, contacts) to answer the question. Further, we distinguish between the time on task before the interruption (**TOT1**) and the time on task after the interruption when participants continued to work on the task (**TOT2**).
- Time of notification: We recorded the time the notification of an incoming call was active (**TN**), i.e. the time from first notification of the call until the conversation started. We also kept track of the time the notification was visible to the user (**TNV**), i.e. the time span a notification was postponed would count for TN but not for TNV. TNV ends when the postpone button is pressed.
- Time on call (**TOC**): We measured how long the people took for the word creation task on the phone, i.e. the duration of the phone call. All time measures had an accuracy of milliseconds.
- We administered a NASA-TLX [11] after each part to assess the participants' workload. In addition (as in [1]), we asked people how annoying the phone call interruptions were, and we asked how respectful the phone application was according to the interruptions during the tasks. All these measures were on the same 20-point scale.

The independent variables were the three conditions of baseline UI, postpone UI and multiplex UI, whereas the dependent variables were the measures explained previously. In total we collected 720 data points for each time-related measure (12 participants x 20 tasks x 3 conditions), and we averaged each user's measures over the 20 trials per condition. We collected 36 data points (12 participants x 3 conditions) for the TLX-related measure.

Results of Study I

We were mainly interested in whether the different UIs had an impact on the workload and task performance time. Effects of the conditions were analyzed using one-way within-subject ANAOVA (with Mauchly's sphericity test satisfied) and post-hoc analysis (with Bonferroni correction).

Impact on Qualitative Measures

The most important results can be drawn from users' feedback on the three UIs. Figure 4 shows the results of the answers people gave regarding the paper-based questionnaires for the three UIs we tested.

- *Mental demand:* The UI condition impacted the mental demand of tasks ($F(2,22)=4.22$, $p<.05$). We can see that the multiplex UI ($M=11.33$) was mentally significantly less demanding than the baseline UI ($M=14.17$, $p<.05$). The postpone UI was in between ($M=13.17$).
- *Effort:* The UI condition also impacted our participants' reported effort ($F(2,22)=5.93$, $p<.01$). We can see that our participants needed significantly less effort for finishing the tasks with the multiplex UI ($M=11.58$) compared to the baseline UI ($M=13.83$, $p<.05$).
- *Frustration:* We also found a significant impact of the UI condition on the measured frustration ($F(2,22)=16.48$, $p<.001$). With the multiplex UI ($M=7.08$) participants were significantly less frustrated than with the postpone ($M=10.42$, $p<.05$) and baseline UI ($M=13.25$, $p<.001$).

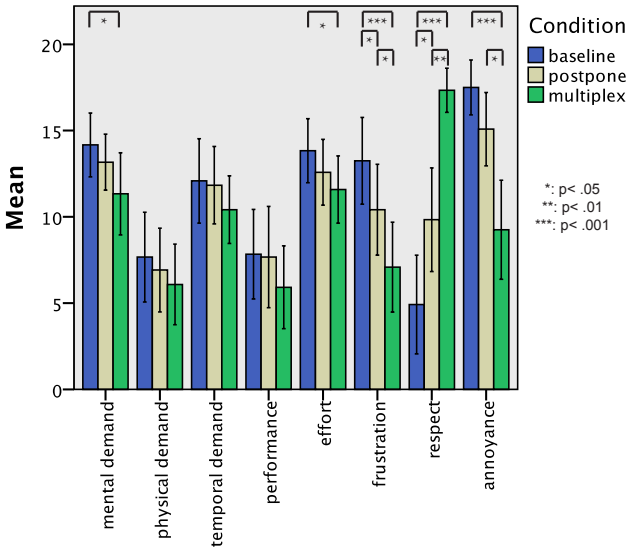


Figure 4. Impact of the UI condition on the measures of NASA-TLX and the measures of respect and annoyance (95% error bars).

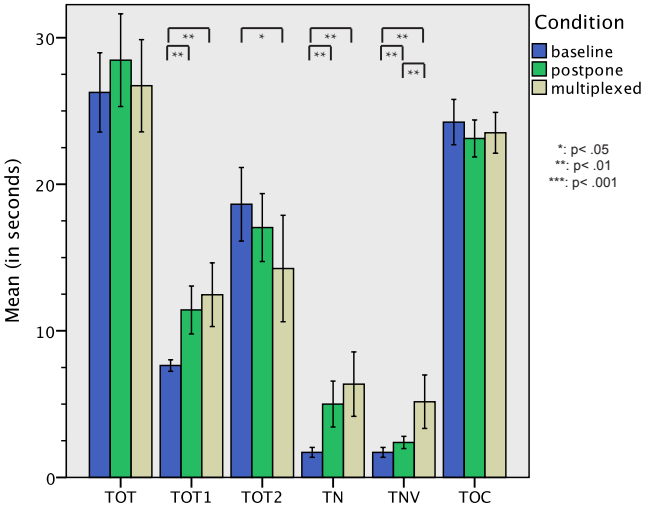


Figure 5. Measured times in seconds (95% error bars).

- *Respect:* We also found a significant impact of the UI condition on the perceived respectfulness ($F(2,22)=35.00$, $p<.001$). When interruptions appeared, our participants found the baseline UI ($M=4.92$) significantly less respectful than the postpone UI ($M=9.83$, $p<.05$) and the multiplex UI ($M=17.33$, $p<.001$). The postpone UI was seen as significantly less respectful than the multiplex UI ($p<.01$).
- *Annoyance*: The UI condition also significantly impacted how annoying an interruption was perceived as being ($F(2,22)=16.94$, $p<.001$). The multiplex UI ($M=9.25$) was significantly less annoying than the baseline ($M=17.50$, $p<.001$), and significantly less annoying than the postpone UI ($M=15.08$, $p<.05$).

These measures show that the multiplex UI allows the users to solve their primary tasks with less mental demand and less effort. Further, for the multiplex UI the users reported

less frustration and less annoyance, and perceived the interruption to be more respectful to their primary tasks.

Impact on Time Measures

On average people took 42.45s (SD=15.75s) to complete one trial consisting of the primary and secondary task. The time on task for non-interrupted trials (M=16.66s) was significantly lower than for interrupted trials (M=27.15s, $t(11)=12.88$, $p<.001$), which replicates earlier findings that the interruption indeed introduces an overhead [16].

We analyzed the impact of the call interruptions on the participant's performance in terms of speed and errors. Figure 5 shows the data for this analysis. We did not find differences in the overall performance (resumption lag, speed and errors) between the three conditions. Participants were a little slower regarding performance time when using the postpone UI. This can be explained by the repetitive opening of the notification when postponing it.

Looking into when people allow for the interruption, we found that the UI condition had a significant effect on the time (TOT1) participants spent with the primary task before the call was accepted ($F(2,22)=15.80$, $p<.001$). For the baseline UI, people on average spent significantly less time (M=7.63s) with the task before the interruption than with the postpone UI (M=11.42s, $p<.01$) and the multiplex UI (M=12.47s, $p<.01$). Consequently, the UI condition also has an significant effect on (TOT2) the time participants spent on the task after the call ($F(2,22)=6.29$, $p<.01$). However, only the difference between the baseline UI (M=18.64s) and the multiplex UI (M=14.25s, $p<.05$) is significant here.

The UI condition has a significant effect on call notification time TN ($F(2,22)=17.29$, $p<.001$). The notification time of calls in the baseline condition (M=1.70s) was significantly shorter than for the postpone UI (M=5.00s, $p<.01$) and the multiplexed UI (M=6.37s, $p<.01$). For the time the call notification was shown to the user we can also find a significant effect of UI condition ($F(2,22)=15.43$, $p<.001$). For the baseline UI, the TNV equals the TN since it cannot run in parallel to the primary task, but it is significantly smaller than for the postpone UI (M=23.82s, $p<.01$) and the multiplex UI (M=51.63s, $p<.01$). More interestingly, the notification with multiplex UI was shown longer than for the postpone UI ($p<.01$).

These findings suggest that people actively used the multiplex UI to display the call notification in parallel while working on the task: they used it to defer the interruption.

With the multiplex UI people used the postpone option less frequently per call (0.26 times mean) than for the postpone UI (0.50 times mean). Though insignificant, this tendency suggests that the multiplex UI decreases the value of the postpone option, since the multiplex UI already makes it possible to continue working on the primary task.

Application Switching Behavior

The UI condition has a significant effect on the usage of the four apps required for answering the task's question ($F(2,22)=6.725$, $p<.01$). With the baseline UI people launched required apps 6.74 times (SD=0.40), with the postpone UI they launched the apps more often (7.09 times, SD=0.68), and with the multiplex UI most often — 7.44 times (SD=0.58). The latter is significantly higher than with the postpone condition ($p<.01$).

Resuming from an interrupted task is a reconstruction process [20], which requires re-opening the apps in our study. While participants switched more often between apps with the multiplex UI, we did not find any significant differences in the usage time of the apps and the total time on the primary task (TOT); i.e., app switching frequency was highest for the multiplexed UI. This suggests that solving tasks requiring more than one app is easier with the multiplexed UI than with the two other UI designs. This is because the multiplex UI does not interfere with the app switching itself, while the baseline and postpone UIs disturb app transitions. As a consequence, the multiplex UI allows for better task reconstruction when re-opening apps is required.

Discussion Study I

Our participants told us that for the postpone UI they would like to have an option for getting back to a call notification in the postpone state instead of waiting until it comes back automatically. This would also allow users to immediately turn their attention towards the call after reaching an intermediate state or finishing a subtask in the primary app. Considering the small effect that for the postpone UI the time on task was longest and the time on call was shortest (see TOT and TOC in Figure 5), it seems like people used the waiting time in the primary task (waiting for the notification to return from being postponed) to prepare for the call task. In contrast, with the multiplex UI one retains control over the call notification and can immediately accept the call if the primary task (or subtask) is completed.

In the lab study, we did not find any significant effect of the UI design on the time participants needed to finish their tasks. Although this study provides us with insights into the differences between the design solutions that we proposed, this study is limited in that the interruptive call is simulated and the tasks are artificial. Since we did not want to overstrain our participants (which might have led to fatigue; sessions already took about 60 minutes) their tasks were rather short. Further, we enticed them to always accept each incoming call. To understand how people would use the multiplex UI in a natural context, we conducted a second study in the wild. For the reasons explained above we decided to choose and implement the multiplex UI for further investigations.

USER STUDY II: AN IN-THE-WILD INVESTIGATION

We extended the prototype of our lab study to a market-ready app for end-users called *CallHeads*. We enhanced the multiplex design of the call notification and implemented the call notification as a circular widget that would show the caller's contact picture and name below: see Figure 4.

Further, by dragging this widget to the screen's edges, the user can accept (green, right), decline (red, left) or postpone (yellow, bottom) the call (see Figure 4b). The colored edges also each show an icon and appear as soon as the user touches the widget. The postpone duration for *CallHeads* was 5 seconds by default and users were able to change it.

Design of Study II

Our second study is designed as a natural experiment [22]: neither we as the researchers nor the study participants had control over the incoming calls and people's tasks. The call interruptions and the tasks the users were currently carrying out on their phones were subject to their natural contexts.

We released *CallHeads* on the Google Play app store so that people could install it to their devices and use the multiplex UI. To study the usage of *CallHeads* we released a second app. Instead of implementing the study within *CallHeads* (which would have required permission to access the Internet), we decided to release a second distinct app for running the study. This allowed users to use the original app without taking part in our study. We believe that this is an important step for making it very transparent to the user that by installing the second app they will take part in a research study (since this was the sole and clearly-stated purpose of the second app). In addition, within the study app we asked participants for consent to take part in the study following the *two-buttons approach* [18]. We did not collect any qualitative measures that we surveyed from participants of Study I. Study II was fully anonymous and we did not collect any data that would disclose anyone's identity or content of conversations.

Results of Study II

We analyzed the data set to characterize the behavior of people when they were called. For investigating differences between groups, we conducted paired t-tests on subsets of the sample containing both conditions per user.

Characteristics of Collected Data

We released *CallHeads* publicly on July 4, 2013. More than 32,000 users downloaded it within 10 weeks and about 10,500 users had it actively installed at the time of writing. 652 of those users agreed to submit data for our study. We withdrew the first two days of data for every user to remove possible self-tests with the app. To purge unnatural user behavior, we further only considered users providing data for more than two days. As such, our final cleaned data set comprises 525 users with data for 31.03 days (SD = 13.86d) per user on average. During this time we observed 88,516 incoming calls, 160.05 per user on average (*SD* = 227.51) and 3.3 calls on average per user and day. 28,906 calls came in while the user had his device unlocked, with 54.36 per user (*SD* = 84.82). These 32.66% of calls constitute interruptions of concurrent app usage that we are interested in, and we limit our further analysis to them. This high number of call interruptions substantiates this as a practical problem for current smartphone usage.

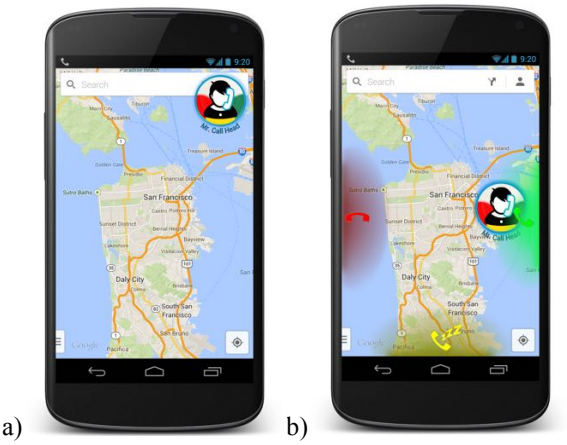


Figure 6. Screenshots of the phone call app deployed on the Google Play Market with a) call notification and b) user interacting with the notification widget to accept a call.

	#cases	*#users*	*per user*
Incoming calls total	**88,516**	**525**	**168**
... non-interruptive	**59,608**	**519**	**114**
... interruptive	**28,908**	**525**	**54**
Interruptive calls accepted	**16,119**	**509**	**31**
...after being postponed	**106**	**79**	**1**
Interruptive calls declined	**2,311**	**317**	**7**
...after being postponed	**114**	**78**	**1**
Interr. calls unanswered	**10,476**	**468**	**149**
...after being postponed	**539**	**206**	**2**
Postpone events	**770**	**247**	**3**
Widget move events	**3,048**	**403**	**7**

Table 1. Descriptive stats on number of calls and events.

Cases of Call Handling

In our uncontrolled study we can observe three kinds of call endings: accepted calls (the user answers the call), declined calls (the user declines the call), and unanswered calls (the caller hangs up or is answered by voicemail). Table 1 provides an overview of the occurrences of the following cases of interruptive calls:

- *Accepted calls*: Out of all interruptive calls more than half (16,119; 55.77%) were accepted, i.e. 32.64 (SD = 49.89) per user on average.
- *Declined calls*: Out of all interruptive calls 2,311 (7.99%) were declined, on average 7.36 (SD = 13.56) per user. In this case the callee actively refuses the call by dragging the widget to the red area (Figure 4). A call can be declined for various reasons [19]. One simple explanation is that the callee does not want to start talking to the caller, or does not want to be interrupted from his current app.
- *Unanswered calls*: Out of all interruptive calls 10,476 (36.24%) went unanswered, i.e. 4.22 (SD = 19.89) per user on average. Note that in these cases of unanswered calls, the phone was not in standby and the user was likely to be using his phone. When a certain time limit is reached, the caller might hang up or the call might possibly be intercepted by the callee's voicemail. This time limit can be reached if the callee repeatedly makes use of

the postpone function or keeps the notification's widget active on screen, maybe after moving it to a corner of the screen, while continuing using the concurrent app.

We can see that most calls were answered and only a few were declined. Note that for the 36% unanswered calls the user had his device in active mode, i.e. they were not unanswered due to unavoidable unavailability, but due to intentional or enforced unavailability [19], as the user just let it ring. Actively using the phone while a call alert is being shown but leaving the call unanswered is a new behavior introduced with our design; we call this *passive decline*.

Timing of Call Handling

It is worth mentioning that only our new design allows the user to continue using his app while the call notification is pending before making a decision on how to handle the call. We analyzed the timing of the decisions to understand how participants made use of this new opportunity. This relates to our lab study's time of notification, but in the wild we also saw people not accepting some calls since they showed natural behavior that we did not control. Again, we can distinguish three cases:

- *Time until accept*: Before a call was accepted, its notification time was 7.08 seconds on average (SD = 6.07s). The reason for the relatively long waiting time is that in this case the callee can also postpone the call or move the call icon out of the focused area. 106 calls (0.65% of accepted) were postponed at least once before being accepted.
- *Time until decline*: The notification time before declining a call was 11.06 seconds on average (SD = 4.41s). This time is 1.65 times higher than for accepted calls. This is a significant difference (t-test on subset of paired cases; $t(307) = 2.14$, $p < .05$). 114 calls (4.93% of declined) were postponed at least once before being declined.
- *Time until ending unanswered*: On average, the notification time for an unanswered call was 23.35 seconds (SD = 18.66s) before the phone stopped ringing. This is significantly higher than the notification time before accepting a call (t-test on subset of paired cases; $t(437) = 13.80$, $p < .001$). 539 calls (5.15% of unanswered) were postponed at least once before being left unanswered.

We can see that for calls resulting in declines, notifications last longer. For calls where the callee instead makes no decision and waits for the caller to hang up, or the voicemail answers, the notification time is even longer. Possibly the notification time is longer when declining since declining a call might be a more cumbersome decision. Further temporizing this decision results in unanswered calls, where the user also might want to pretend unavailability.

Usage of Postpone

In total we recorded 770 postpone events; 197 were sequences of postpone events, i.e. cases where the callee postponed the same call more than once. Nearly half of all users (47.05%) postponed a call at least once; on average a user postponed 4.46 times (*SD* = 6.08). And on average 2.66% of calls were postponed at least once.

Interestingly, 499 cases (64.81%) of postpone actions led to unanswered calls. As already mentioned, this can either result from the callee having voicemail, or the caller being unwilling to wait any longer and hanging up. Looking at these nearly two-thirds of postpone cases, we found that for 23.64% of calls the caller was willing to wait for more than 40s; average waiting time was 30.22s (*SD* = 13.31s).

We expected the postpone option to be used more often than only in 2.66% of calls. This underpins that postpone is not essential for mitigating interruptions when using a multiplex UI, but it can be helpful in certain contexts.

App Usage with Notification Being Multiplexed

We also investigated whether the user's call handling is influenced by the app which is being interrupted. Therefore we looked for apps that we observed to frequently be interrupted when a phone call comes in. We found that the likelihood of using the postpone option is high when media applications are being used. The probability that a call interrupting the app *"MX Player"* will be postponed is 0.24 (17 interr. calls), and for *YouTube* 0.23 (139 interr. calls). In contrast, interruptions of apps that belong to the communications category have a lower probability that the call will be postponed. For instance, for the contact book, the MMS application and *WhatsApp* the probabilities of calls being postponed are 0.03 (1,779 interr. calls), 0.04 (978 interr. calls), and 0.06 (1,409 interr. calls) respectively.

Discussion of Study II

The study reveals that one-third of incoming phone calls interrupt concurrently-used apps. This emphasizes the need to improve UIs for handling phone calls, since these interruptions introduce a significant overhead [16].

Analyzing the use of the postpone function, we found that users leverage it to passively decline calls even though they are using their phones. So far, without this function the phone could not be used, and app usage could not be continued, until the call was either accepted, declined or left unanswered. Since the caller does not know when the call is postponed, the caller has the feeling that the callee is not available (e.g. away from the phone). This is only possible through our new design.

Further Refinements of the Design

Resulting from 32,000 installations of *CallHeads*, we also received valuable feedback through both the app store's comment function as well as by email. The most requested feature that people would like to use is to be able to drag-and-drop the widget to an additional area for declining and sending prewritten text messages to the caller. This is interesting, since we found that it takes longer to decline a call than to accept a call. Providing users with an option to inform the caller about the reason why they were declined might improve this decision-making. In addition, some of those calls that ended unanswered might have been declined with an explanation provided to the caller instead of pretending unavailability [19]. Further, it might also be valua-

ble to provide a function to accept a call in speaker-phone mode, so that after accepting the call the user can stay in the app and start talking directly and hands-free to the caller. This would further allow for multitasking between the app usage and the conversation, but also has other implications (e.g. people nearby can listen to the conversation).

Limitations

Our in-the-wild study is limited by the inherent properties of the method of running large-scale studies through the app store. Most importantly, we cannot know about the user's context when calls came in. For a better understanding of the reasons why a call ended unanswered even though the user was on his phone, we plan to enhance the quantitative study presented in this paper with qualitative methods of experience sampling in the large. Further, we do not know anything about the relation between caller and callee; this might impact how the callee handles the call.

DISCUSSION AND FUTURE WORK

With *Study I* we found that the multiplex UI is best suited for handling call interruptions, and with *Study II* we analyzed how people use the multiplex UI in natural contexts. We also need to consider the caller and discuss effects we could not reveal in our studies.

Keeping the Caller Waiting on the Line

When the receiver postpones a call the caller will be kept waiting on the line, and as we found this might result in an unanswered call. Since related work [4,25] found that it has a positive effect when the caller is aware of the callee's status, one idea might be to signal the caller as to what is happening. One possibility could be the design of special call-progress tones, or we could use speech synthesis to signal the callee's current app usage context to the user, e.g. *"the person you are calling is currently playing Angry Birds"*. Then the caller might be able to make a more informed decision about how long he wants to wait for the callee to pick up the call. For future work we plan to signal the callee's context back to the caller to investigate how the caller reacts if he knows that the other person is using his phone, but does not want to start a conversation; issues of privacy and social aspects also need to be considered.

Social Implications

The current design of phone call UIs also has some social implications: current implementations on the different operating systems force the user to either accept or decline an incoming call if he wants to continue using his device. Otherwise he would have to wait until the voicemail answers or the caller hangs up. If the user wants to keep using his current app, he will have to decline the call if he wants to evade the interruption. As a result, declining the call might have an impact on the peers' social relation. Our new design allows the user to pretend unavailability while continuing to use apps on his smartphone. Future work will study the effect of this opportunity on people's phone communication behavior leveraging the *CallHeads* deployment.

Switching to Call after App Episodes

The multiplex UI design we propose allows people to continue using their apps while being notified of the phone call in parallel. Both studies provide evidence that people make use of this new functionality to defer the interruption for a short time to finish micro-interactions with their apps. In particular, this form of pre-alerting for incoming calls allows people to finish their current episode of interactions with the current app, before they allow the interruption [8]. We saw in Study I that people kept the notification open for a certain time that they would need to reach an intermediate state in their tasks before they decided to switch to the call.

Change Blindness and Call Blindness

When adding dynamic visual content to the display one has to be concerned about the effect of change blindness [7]: the popup of the notification might result in the user missing changes within the primary app. This effect is greatest when the popup opens full-screen, as with current phone call apps. However, on the other hand, if the notification is made smaller or placed less prominently, it might not be noticed by a user engaging with other applications, which might result in the user not recognizing the call. In fact, we got requests from users of the *CallHeads* app to be able to change the appearance of the notification (e.g. increase the size of the font and the widget itself). However, in the *CallHeads* app the call was also signaled by the ringtone and vibration (if configured) as long as the user did not react to the notification (by touching the widget). However, every one of our users receiving interruptive calls interacted with the widget at least once, so our users did not miss it. Nonetheless, finding the optimal size for the notification, i.e. the ratio of multiplexing between the primary app and the notification, is a subject for future work.

Other Modalities

Our design considerations target visual attention. In addition, incoming calls are also announced by auditory and haptic signals. A holistic design needs to consider these modalities to notify the user of incoming phone calls. One possibility could be changing the ringtone to unobtrusive sounds. Hence, the user could be notified about an incoming phone call in an ambient way. Also, one could apply different vibration patterns to create haptic notifications in accord with the visual notification. The integration of different modalities therefore needs to be addressed in future work, and should be aligned with the visual notification.

The Non-interruptive Case

While the proposed multiplex UI is dedicated to the interruptive case, where a user is engaged with another task on the mobile device, we have to raise the question of how to proceed for the 67% of non-interruptive calls (i.e., the phone is in standby mode when a call comes in). *CallHeads* was built in such a way that it does not show up in this case, and instead the default phone call appears. Another option would be to apply the multiplex UI that we proposed for the interruptive case. The problem, however, is that there is no primary task on the phone when the phone is not being

used, though one could consider the device's lock-screen as the in-use app. From our users we got strong feedback that they would also like to use the multiplex UI and the option to postpone a call in the non-interruptive case. This could support users engaged in non-phone tasks when calls come in. This would instantly mute the phone (similar to silencing) and the notification would come back after a few seconds. Postponing a call might be beneficial when one has to leave the room before being able to answer a call.

CONCLUSION

This paper presents a multiplex UI for handling incoming calls on smartphones. This design solution tackles the problem that calls can interrupt concurrent application usage. We revisited the current design of phone call UIs, extended the options for handling incoming phone calls and presented considerations for possibilities to postpone calls and multiplex the call notification with the concurrent app. We studied these two proposals for the design of phone call apps in a small-scale controlled lab study. We found that the multiplex UI improves call handling with concurrent app use, in particular because it is less frustrating and annoying. We also released an implementation of the multiplex UI to more than 32,000 users through a commercial app store. Some of these users (525) contributed to a study to understand how the app was used in the wild. Results showed that one-third of incoming calls interrupt concurrent app usage, and that people use the postpone option to continue using their apps, often leaving their call unanswered. This was not possible with previous phone call UIs.

ACKNOWLEDGEMENT

We thank Jessica Cauchard and Markus Löchtefeld for providing us their task designs [6].

REFERENCES

1. Adamczyk, P.D., Bailey, B.P. If not now, when?: the effects of interruption at different moments within task execution. In Proc. of CHI '04
2. Alt, F., Shirazi, A. S., Schmidt, A., Atterer, R. Bridging waiting times on web pages. In Proc. of MobileHCI '12.
3. Altmann, E.M., Trafton, J.G. Memory for goals: an activation-based model. Cog. Sci., 26, (2002), 39-83.
4. Avrahami, D., Gergle, D., Hudson, S. E., Kiesler, S. Improving the match between callers and receivers: a study on the effect of contextual information on cell phone interruptions. Behav. Inf. Tech. 26, 3 (2007).
5. Bogunovich, P., Salvucci, D.D. The effects of time constraints on user behavior for deferrable interruptions. In Proc. of CHI '11.
6. Cauchard, J., Löchtefeld, M., Fraser, M., Krüger, A., Subramanian, S. m+pSpaces: virtual workspaces in the spatially-aware mobile environment. In MobileHCI '12
7. Davies, T., Beeharee, A. The case of the missed icon: change blindness on mobile devices. In Proc. of CHI '12
8. Fischer, J. E., Greenhalgh, C., Benford, S. Investigating episodes of mobile phone activity as indicators of opportune moments to deliver notifications. In Proc. of MobileHCI '11.
9. González, V.M., Mark, G. Constant, constant, multi-tasking craziness: managing multiple working spheres. In Proc. of CHI '04.
10. Grandhi, S. A., Schuler, R., Jones, Q. G. Telling calls: facilitating mobile phone conversation grounding and management. In Proc. of CHI 2011.
11. Hart, S., Staveland, L. Development of NASA-TLX (Task Load Index): results of empirical and theoretical research. In Human Mental Workload. 1988.
12. Ho, J., Intille, S. S. Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In Proc. of CHI '05.
13. Iqbal, S.T., Horvitz, E., Ju, Y. C., Mathews, E. Hang on a sec!: effects of proactive mediation of phone conversations while driving. In Proc. of CHI '11.
14. Iqbal, S.T., Bailey, B.P. Oasis: a framework for linking notification delivery to the perceptual structure of goal-directed tasks. ACM ToCHI, 17, (2010), 15.
15. Knittel, J., Shirazi, A. S., Henze, N., Schmidt, A. Utilizing contextual information for mobile communication. In Proc. of CHI EA '13.
16. Leiva, L., Böhmer, M., Gehring, S., Krüger, A. Back to the app: the costs of mobile application interruptions. In Proc. of MobileHCI '12.
17. Miyata, Y., Norman, D.A., Psychological issues in support of multiple activities. In User Centered System Design: New Perspectives on HCI. (1986), 265-284.
18. Pielot, M., Henze, N., Boll, S. Experiments in app stores - how to ask users for their consent? In CHI'11 Workshop on Ethics, Logs and Videotape.
19. Salovaara, A., Lindqvist, A., Hasu, T., Häkkilä, J. The phone rings but the user doesn't answer: unavailability in mobile communication. In Proc. MobileHCI 2011.
20. Salvucci, D.D., Bogunovich, P. Multitasking and monotasking: the effects of mental workload on deferred task interruptions. In Proc. CHI 2010.
21. Schmidt, A., Takaluoma, A., Mäntyjärvi, J. Context-aware telephony over WAP. Pers. Ubi. Computing, 4(4):225–229, 2000.
22. Shadish, W. R., Cook, T. D., Campbell, D. T. Experimental and Quasi-Experimental Designs for Generalized Causal Inference. Cengage Learning, 2ed, 2001.
23. Stamm, K., Ahamed, S. I., Madiraju, P., Zulkernain, S. Mobile intelligent interruptions management (MIIM): a context aware unavailability system. In Proc. SAC '10.
24. Strayer, D. L., Johnston, W. A. 2001. Driven to distraction: dual-task studies of simulated driving and conversing on a cellular telephone. Psych. Sci. 12, 462-466.
25. Teevan, J., Hehmeyer, A. Understanding how the projection of availability state impacts the reception incoming communication. In Proc. of CSCW '13.
26. ter Hofte, H. Xensible interruptions from your mobile phone. In Proc. of MobileHCI '07.
27. Trafton, J.G., Monk, C.M., Task interruptions. In Reviews of Hum. Factors and Ergo., 3 (2007), 111-126.
28. Want, R. When cell phones become computers. IEEE Pervasive Computing, 8(2):2–5, 2009.