Context-aware Communication in the Car

Tobias Martin
University of Munich (LMU)
Munich, Germany
t.martin@campus.lmu.de

Stefan Lang
University of Munich (LMU)
Munich, Germany
stefan.lang@campus.lmu.de

Simon Weiser
University of Munich (LMU)
Munich, Germany
simon.weiser@campus.lmu.de

ABSTRACT

How to reduce distraction by a smartphone while driving is a common question nowadays. The number of mobile phone users is increasing enormously. And concurrently increases the desire of being available - no matter if it's a situation you shouldn't be distracted or even in danger situations. Due to this desire, the callee should not decide whether to be called but the caller. Here we describe a concept and a implementation to reduce the communication in the car - especially in danger situations. We designed an app that hands over the responsibility about, if a call should be executed, to the initiator by displaying informations about the context of the contact. The final user study confirmed our approach and the context information is perceived correctly.

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation (e.g., HCI): User Interfaces — *Prototyping*; H.4.3 Information Systems Applications: Communications Applications

Author Keywords

Automotive user interfaces, calling while driving, context sharing, driving safety, phone call.

INTRODUCTION

There are about 4.5 to 5 billion mobile phone users worldwide [6] and about 80 % of all drivers are owning a smartphone.

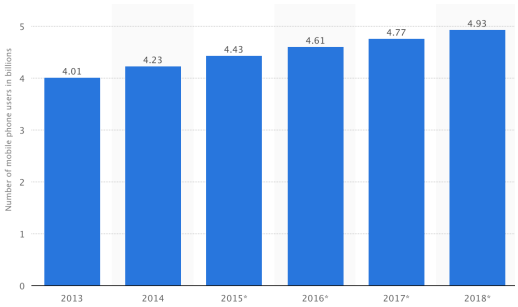


Figure 1: Mobilephone users worldwide [6]

The extensive use of smartphones in the daily life produces a huge desire to be "always on". This desire leads people to want to be available, in situations they should better take care about their environment, e.g. when driving a vehicle in bad weather conditions. The risk of collision is four times higher during cell phone use while driving [1].

Therefore the responsibility to execute a call should be on the side of the initiator. But in order to give the initiator the responsibility, there is the need to transfer context information about the callee.

Context information could be:

- *In Vehicle*
  If the contact is driving or not.
- *Vehicle Type*
  The type of the vehicle, e.g. car or bicycle.
- *Speed*
  The speed with which the contact is on the road.
- *Weather Conditions*
  The weather conditions at the location of the contact. Is it sunny or raining?
- *Geo Location*
  The exact position or the approximate region.
- *Speakerphone*
  If hands-free speaking is possible through speakerphone in the vehicle.
- *And many more*

In the last decade there was a lot of research about this topic. Following we will point some related work to show the wide field of the current research as well as our concept and the implementation of the context-call Android application we designed. After building the application we conducted a user study to prove our expectation.

RELATED WORK

In 2007 Kern et al. evaluated an approach similar to ours [4]. They designed a system that informs the caller about the callee's context *before* initiating the call. Their goal was to thereby avoid unnecessary calls that would have ended anyway as soon as the caller knows the callee is in a driving situation. Their prototype "InCA-System" distinguished between three abstract recommendations for the caller:

1. **Green**: Call can be taken (without any concerns)

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

2. **Yellow**: Call would be inopportune (only call if necessary)
3. **Red**: Call undesired by the callee (only call if very urgent)

The latter of these statuses can only be set manually by the driver, whereas the first two are automatically detected through in-car sensors. The evaluation of their system showed that the recommendations to the callers were considered and that the systems would be used by the drivers to make their driving experience more relaxing.

DeDe is another approach to define context-aware communication. Y. Jung, P. Persson and J. Blom present in their paper [7] the design, implementation and validation of an enhanced mobile phone messaging system (DeDe), allowing the sender to define the context in which the message will be deliverd to the recipient.

In Pittsburgh they researched in 2002 [5] a signaling method to mitigate the risk of automobile accidents with in-car cell phone use. They show in a experiment how remote cell phone callers were induced to speak less during critical driving periods and in a second experiment, how the performance of a driver improved when callers reduced talking levels during critical driving situations. »»»> origin/master

CONCEPT

The goal was to build a simple tool which fits straight into the app-infrastructure of a usual smartphone user. We decided to replace the default contacts list app with our implementation of a contacts list app.

The gap that our app should close is the transfer of the context information. With this app the initiator is able to see some relevant information before the call happens, which help to decide if the call is important enough to execute under the given circumstances of the callee.

To deliver the context information from person A to person B the most efficient way is the following: person A sends the context information to a server every time the context changes. Person B requests the data from the server when it's required, e.g. when the contact is opened in the contacts list app (see figure 2).

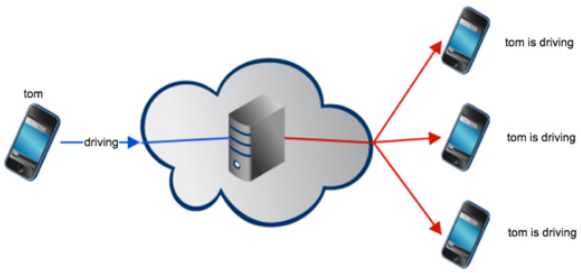


Figure 2: Concept

IMPLEMENTATION

We designed a simple and ordinary contacts app which is enhanced by displaying the context of the person you want to call. In this case and for our goal, context refers particularly to the following information:

Context Information
Activity
Road Type
Destination
Remaining Travel Time
Position
Hands-free Speaking
Speed
Weather

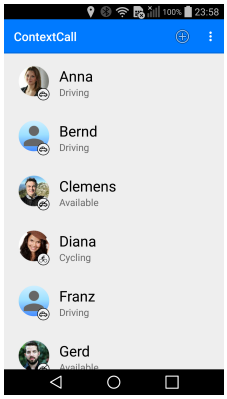
Table 1: Context Information.

Firstly, one will want to know what the callee is doing at the moment. We distinguished between the activities *driving*, *cycling* and *still* whereby the latter means the recipient of the call is doing nothing right now and his/her device is motionless, e.g. on a table etc. Depending on if someone is driving you want to have the listed additional information. One would be the road type which lets you know if someone is driving e.g. in a city or on a highway etc. Another one is the destination the person is heading to and the remaining travel time. Therefore you need the person's current position. This could either be a pinpoint GPS-coordinate, or if the user does not want to share his exact location, just a radius or a place name of his current position. Another very important aspect of the person's context is if hands-free speaking is enabled or disabled while in the car.

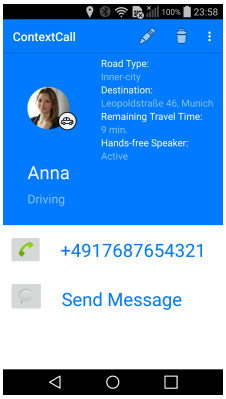
Furthermore, it is helpful to have data about the speed and the current weather situation. At the recent state of our app all of these information can be accessed automatically except the destination and the remaining travel time, which have to be typed in manually.

App Design

The Android-App we built, basically consists of two different views, an overview and a detail view (see figure 3).



(a) Overview



(b) Detailview

Figure 3: Android Application ContextCall

On the overview on the left you see all your contacts as you know it from other contacts apps, added with information about a personfs activity, which is shown by the icons next to the image and the string below the name. By tapping on a contact, the detail view appears and shows the additional context information mentioned in table 1. The particular case in figure 3b for example gives information about the road type, the destination and the remaining travel time. Furthermore you get informed that hands-free speaking is enabled. If you then decide to call 'Anna' although she is driving, a small alert pops up and gives you three options (see figure 4). You can either choose to call her or not, or you make use of the '*remind me*'-option which notifies you when her status is 'still'.

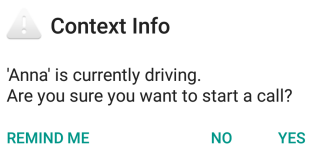


Figure 4: Alert Pop-Up

At the beginning we implemented most of the context recognition by our own. We used GPS for example, to find out if someone is driving. If someone's speed was over 10 kilometers per hour we set the status to 'driving'. But in mid-May of 2016 Google released its new so-called *Awareness API*[1] which made things much easier for us as programmers. From this point on we were able to get the activity data of a user by only a few lines of code and this API became the core of our application.

The Awareness API is part of the *Google Play Services* and is a unified sensing platform, enabling apps to be aware of all aspects of a user's context, while managing system health for you [2]. With this API your app is able to recognize the following 7 different context types [3]:

Location The user's current location as a latitude and longitude value.

Place A semantic version of a location that is called place (including the place type, e.g. a coffee shop).

Beacons What is around a user? Are there nearby beacons that can be detected and identified?

Time The local time of a user that can be combined with other context information to form a more complex condition.

Headphones' State Are headphones plugged in the device or not?

Weather Ambient conditions like the weather, which have an effect on the user's behavior.

Activity The detected user activity (e.g. walking, running, biking and driving)

[1]https://developers.google.com/awareness/

The latter of these is the important one for the goal of activity and context recognition. All of these information can be combined using *AND*, *OR*, and *NOT* boolean operators to build complicated conditions that have to be met to trigger a notification. E.g. you can construct a condition that says the user is driving in the car AND he is near a pharmacy AND it is during the opening hours of that shop. If these requirements are fulfilled, then you tell the user that he can pick up the wanted medication.

In particular, we used the *Fence API*[2], which is part of the Awareness API. The concept of *fences* is taken from Geo-Fencing, in which a geographic region, or "Geo- Fence", is defined, and an app receives callbacks when a user enters or leaves a region. What differs in this case is, that an activity is 'entered', not a region. So whenever the activity state transitions, it lets our app react to the user's current activity. For example: "Tell me whenever the user is driving". Once the conditions are met, the app receives a callback and we can update the status of a user.

USER STUDY

To evaluate our concept and the actual implementation we invited a total of 25 participants to our user study. As the overall goal was to evaluate if users perceive the context information provided by our application correctly, we decided to generate 10 different use-cases (scenarios). We vocally discussed each scenario with the current participant, but all the context information had to be pulled out from the running app on a provided smartphone. Differing by the call trigger (meaning the reason why a participant had to call a person) and the context the person being called (callee) is in, the participants had to decide whether they would like to:

1. Make the call (or)
2. Not make the call (and)
3. Not make the call but send a text message (and/or)
4. Not make the call but get notified when the driver status changes

Following the scenarios we asked additional questions with the participants acting the role of the driver (the person being called). Goal of these questions was to gather inside in the willingness of potential users to share more or less specific information.

The following figures visualise the outcome of our user study. Figure 5 to 7 focus on the ten different scenarios which can be distinguished along the x-axis. As mentioned they differ in the status the callee is in (driving, cycling, still). Distribution of the answers given by our participants can be seen along the y-axis, while colours visualise the four possible choices.

[2]https://developers.google.com/awareness/android-api/fence-api-overview

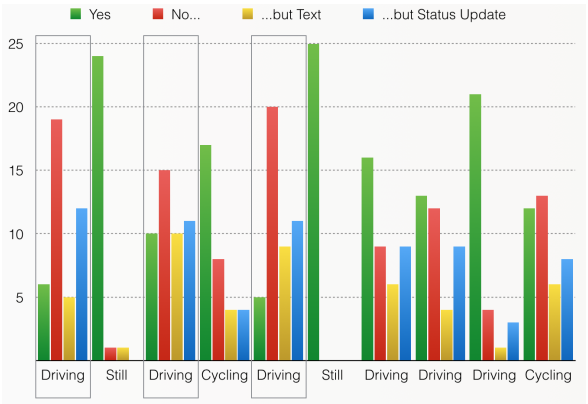


Figure 5: User Study: Would you make the call (status driving)

Highlighted in figure 5 you can see three different scenarios with the callee in the status 'driving'. As anticipated the majority of participants would not make a call in this situation (red bars). Moreover the majority would not like to send a text message straight away, or get notified if the status of the driver changes. Regarding our concept this outcome allows us to suppose that participants pulled the correct context information out of the app and that they consider the safety of the person they want to call.

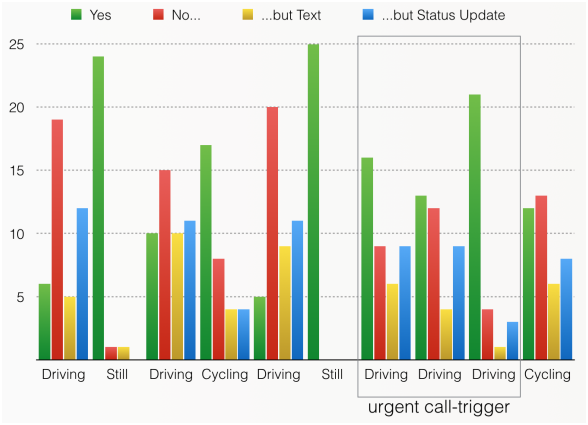


Figure 6: User Study: Would you make the call (status driving/with urgent call-trigger)

Contrary to the confirmed hypotheses shown in figure 5 (people will not make the call when they see the callee is driving), we found three 'driving' scenarios in which the participants would in fact make the call, although the callee is displayed as driving. Figure 6 focuses on these three scenarios with a majority of participants making the call (green bars). Discussion with our participants provided enough feedback to ensure that these decisions were based on the call-trigger (the reason they had to make the call). With call-triggers getting urgent (an emergency in the family for instance), people tend to ignore the context of the callee. Consulting with the participants we decided that users should always have the possibility to make calls in case of emergencies and that blocking calls completely is ineligible.

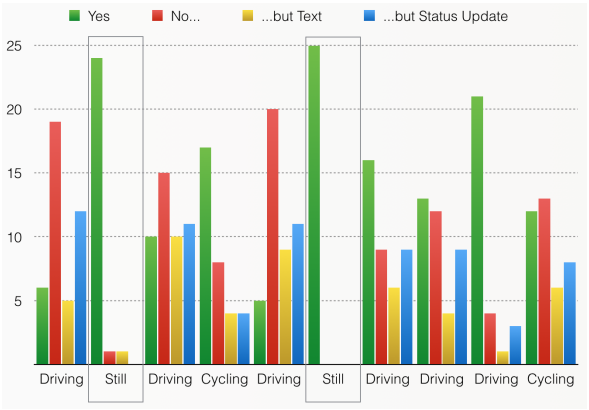


Figure 7: User Study: Would you make the call (status still)

To gain even more proof of our concept we generated two scenarios with the callee being in the status 'still'. Figure 7 focuses on these two scenarios and provides assurance that participants perceive our provided context information correctly and make their call straight away.

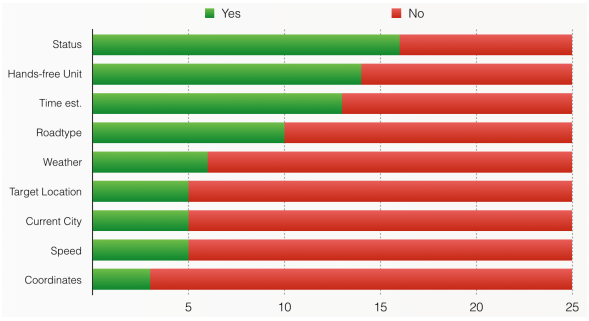


Figure 8: User Study: What information are you willing to share

With figure 8 we try to visualise the willingness of our participants to share context information with their phone contacts. After running through the previous ten scenarios we made sure that the participants understood the importance of the provided information and that more information might lead to a better understanding of the callee status (traffic situation, stress level and so on). Regarding the figure, the y-axis shows different information we would be able to provide in our concept app, the x-axis shows the amount of participants willing (green) or not willing (red) to share this information. As we can see, the willingness to share information is to be considered. While basic user information like the status (driving, cycling, still) or the availability of a hands-free unit is likely to being shared, more specific information like the target location or the precise user coordinates are not want to be published throughout the users contacts. Again consulting with our participants we

found out that the willingness of information sharing increases, if the users were able to decide with information is provided for which contact in their lists. Carrying on this thought, participants ask for a grouping options (for instance 'family', 'friends', 'work') to adjust information sharing by hand.

CONCLUSION

Concluding our user study we can say that the visual processing of the context information provided by our application is perceived correctly. The currently implemented degree of information, as seen in figure 3 is regarded as sufficient to distinguish between the three main statuses driving, cycling and still. In addition we figured out, that the reason why calls are being made might in fact make calls more important than safety concerns with the callee being in a driving situation. Moreover we had interesting insights in data privacy concerns of possible users, which make concept adjustments necessary to improve willingness to share information.

FUTURE WORK

To analyse user behaviour more closely we could think of reworking part of our application so it can be published in the Android app store. As this would exceed our current server capacities we would have to switch back-end strategies, as well as implement an appropriate logging tool to keep track of the generated user data.

REFERENCES

1. Insurance Institute for Highway Safety. 2010. Cell phone use while driving and attributable crash risk. (2010). `http://www.ncbi.nlm.nih.gov/pubmed/20872301` Retrieved August 27, 2016.
2. Google Inc. 2016a. Introducing the Awareness API, an easy way to make your apps context aware - Google I/O 2016. Video. (18 May 2016). Retrieved August 22, 2016 from `https://www.youtube.com/watch?v=37ia7S4Lsv4`.
3. Google Inc. 2016b. What is the Awareness API? (2016). `https://developers.google.com/awareness/overview` Retrieved August 22, 2016.
4. Dagmar Kern, Albrecht Schmidt, Michael Pitz, and Klaus Bengler. 2007. Status- und Kontextinformationen für die Telekommunikation im Auto. In *Mensch und Computer 2007: Interaktion im Plural*, Tom Gross (Ed.). Oldenbourg Verlag, München, 119–128.
5. Mike Schneider Sara Kiesler Dan Siewiorek Punitha Manalavan, Asad Samar. 2002. In-car cell phone use: mitigating risk by signaling remote callers. In *CHI EA '02 CHI '02 Extended Abstracts on Human Factors in Computing Systems*. ACM, New York, 790–791.
6. Statista. 2014. Number of mobile phone users worldwide from 2013 to 2019. (2014). `http://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide` Retrieved August 27, 2016.
7. Jan Blom Younghee Jung, Per Persson. 2005. DeDe: design and evaluation of a context-enhanced mobile messaging system. In *CHI '05 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, 351–360.