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 while driving? This is a common question in these time of being always available. The number of mobile phone users are is increasing enormously. And so increases the desire of being available - no matter if you are in situations you shouldn't be distracted or even in danger situations. Due to this desire, the person called should not decide whether he is going to be called but the caller. Here we describe a solution to reduce the communication in the car - espacially in danger situations. 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.

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 [14] and about 80 % of all drivers are owning a smartphone.

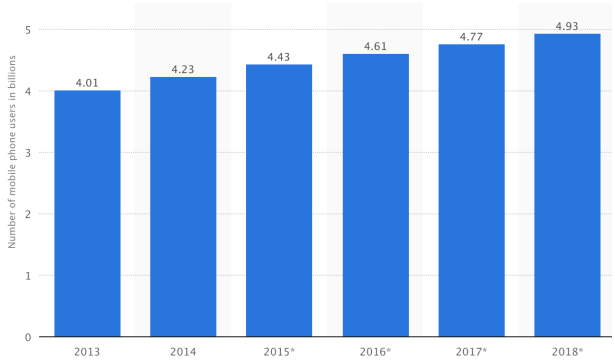


Figure 1: Mobilephone users worldwide [14]

The extensive use of smartphones in the daily life produce 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 u drive a vehicle in bad weather conditions. The risk of collision is four times higher during cell phone use while driving [5].

Because of this, the responsibility to execute a call should be on the side of the initiator. But to give the initiator the responsibility, there is the need to transfer context information about the contact to call.

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.
- *Roadtype*
  If the contact is "In Vehicle" the roadtype could be relevant. Is the contact driving through the city or on a motorway.
- *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*

RELATED WORK

TODO: Some related work here.

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 contactlist-app with our implementation of a contactlist-app.

The gap that our app should close is the transfer of the context information. With this app the initiator sees, before the call happens, some relevant informaion to help to decide if the call is important enough to execute.

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

To deliver the context information from person A to person B the most efficent way is that person A sends the context information to a server everytimes the context changes. Person B requests the data from the server when it's required, e.g. when the contact is opened in the contactlist-app (see figure 2).

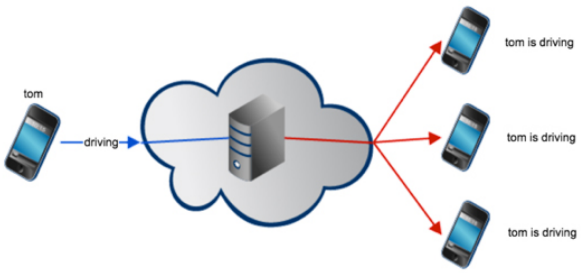


Figure 2: Concept

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

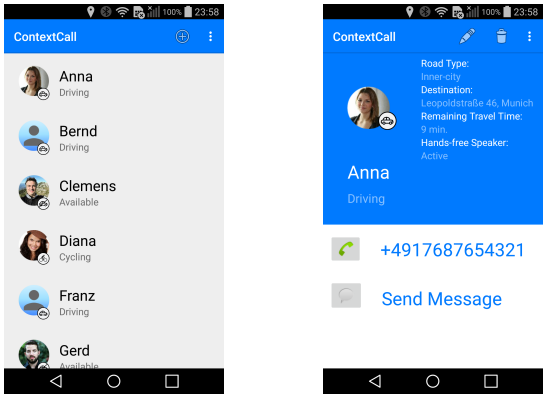
Table 1: Context Information.

Firstly, one will want to know what the recipient of the call is doing at the moment. We distinguished between the activities *driving*, *cycling* and *still* whereby the latter means the person being called 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 person's 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'.

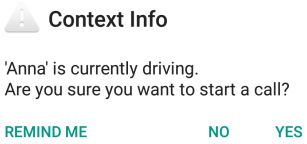


Figure 4: Alert Pop-Up

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

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

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 [7]. With this API your app is able to recognize the following 7 different context types [8]:

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)

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)

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

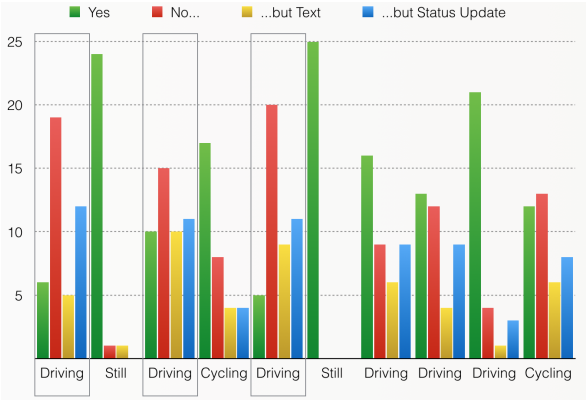


Figure 5: User Study: Would you make the call (different scenarios)

and the context the person being called 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 10 different scenarios which can be distinguished along the x-axis. As mentioned they differ in the status the person being called 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.

Highlighted in figure 5 you can see three different scenarios with the person being calling in the status 'driving'. As anticipated the majority of participants would not make a call in this situation (red bar). 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.

In figure 6

In figure 7

In figure 8

CONCLUSION AND FUTURE WORK

TODO: put text here

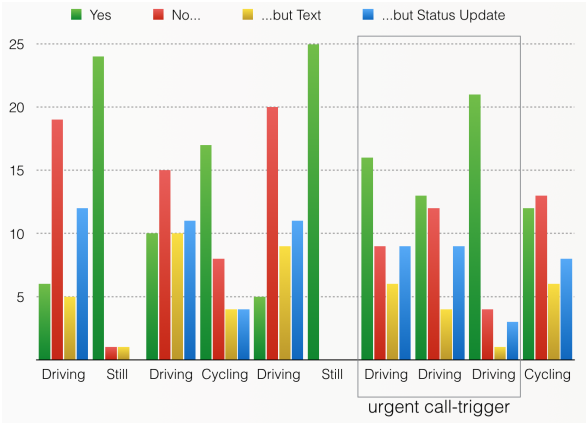


Figure 6: User Study: Would you make the call (different scenarios)

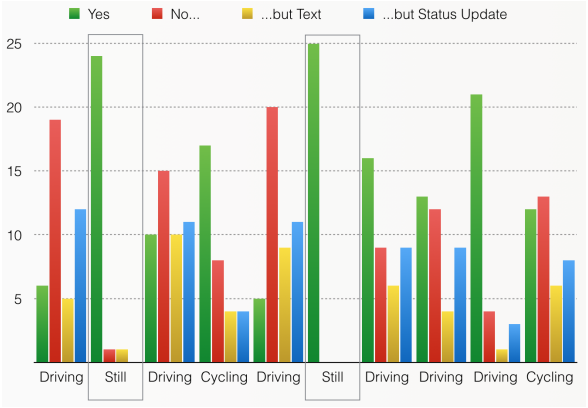


Figure 7: User Study: Would you make the call (different scenarios)

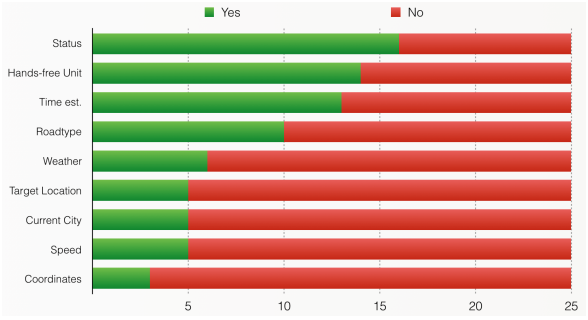


Figure 8: User Study: What information are you willing to share

ACKNOWLEDGMENTS

Sample text: We thank all the volunteers, and all publications support and staff, who wrote and provided helpful comments on previous versions of this document. Authors 1, 2, and 3 gratefully acknowledge the grant from NSF (#1234–2012–ABC). *This whole paragraph is just an example.*

Balancing columns in a ref list is a bit of a pain because you either use a hack like flushend or balance, or manually insert a column break. http://www.tex.ac.uk/cgi-bin/texfaq2html?label=balance multicols doesn't work because we're already in two-column mode, and flushend isn't awesome, so I choose balance. See this for more info: http://cs.brown.edu/system/software/latex/doc/balance.pdf

Note that in a perfect world balance wants to be in the first column of the last page.

If balance doesn't work for you, you can remove that and hard-code a column break into the bbl file right before you submit:

http://stackoverflow.com/questions/2149854/how-to-manually-equalize-columns- in-an-ieee-paper-if-using-bibtex

Or, just remove and give up on balancing the last page.

REFERENCES FORMAT

Your references should be published materials accessible to the public. Internal technical reports may be cited only if they are easily accessible and may be obtained by any reader for a nominal fee. Proprietary information may not be cited. Private communications should be acknowledged in the main text, not referenced (e.g., [Golovchinsky, personal communication]). References must be the same font size as other body text. References should be in alphabetical order by last name of first author. Use a numbered list of references at the end of the article, ordered alphabetically by last name of first author, and referenced by numbers in brackets. For papers from conference proceedings, include the title of the paper and the name of the conference. Do not include the location of the conference or the exact date; do include the page numbers if available.

References should be in ACM citation format: `http://www.acm.org/publications/submissions/latex_style`. This includes citations to Internet resources [4, 3, 12] according to ACM format, although it is often appropriate to include URLs directly in the text, as above. Example reference formatting for individual journal articles [2], articles in conference proceedings [10], books [13], theses [15], book chapters [16], an entire journal issue [9], websites [1, 3], tweets [4], patents [6], games [11], and online videos [12] is given here. See the examples of citations at the end of this document and in the accompanying `BibTeX` document. This formatting is a edited version of the format automatically generated by the ACM Digital Library (`http://dl.acm.org`) as "ACM Ref." DOI and/or URL links are optional but encouraged as are full first names. Note that the Hyperlink style used throughout this document uses blue links; however, URLs in the references section may optionally appear in black.

BALANCE COLUMNS

REFERENCES FORMAT References must be the same font size as other body text.

REFERENCES

1. ACM. 1998. How to Classify Works Using ACM's Computing Classification System. (1998). http://www.acm.org/class/how_to_use.html.
2. R. E. Anderson. 1992. Social Impacts of Computing: Codes of Professional Ethics. *Social Science Computer Review December* 10, 4 (1992), 453–469. DOI: http://dx.doi.org/10.1177/089443939201000402
3. Anna Cavender, Shari Trewin, and Vicki Hanson. 2014. Accessible Writing Guide. (2014). http://www.sigaccess.org/welcome-to-sigaccess/resources/accessible-writing-guide/.
4. @_CHINOSAUR. 2014. "VENUE IS TOO COLD" #BINGO #CHI2014. Tweet. (1 May 2014). Retrieved Febuary 2, 2015 from https://twitter.com/_CHINOSAUR/status/461864317415989248.
5. Insurance Institute for Highway Safety. 2010. Cell phone use while driving and attributable crash risk. (2010). http://www.ncbi.nlm.nih.gov/pubmed/20872301
6. Morton L. Heilig. 1962. Sensorama Simulator. U.S. Patent 3,050,870. (28 August 1962). Filed Februrary 22, 1962.
7. 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.
8. Google Inc. 2016b. What is the Awareness API? (2016). https://developers.google.com/awareness/overview Retrieved August 22, 2016.
9. Jofish Kaye and Paul Dourish. 2014. Special issue on science fiction and ubiquitous computing. *Personal and Ubiquitous Computing* 18, 4 (2014), 765–766. DOI: http://dx.doi.org/10.1007/s00779-014-0773-4
10. Scott R. Klemmer, Michael Thomsen, Ethan Phelps-Goodman, Robert Lee, and James A. Landay. 2002. Where Do Web Sites Come from?: Capturing and Interacting with Design History. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '02)*. ACM, New York, NY, USA, 1–8. DOI:http://dx.doi.org/10.1145/503376.503378
11. Nintendo R&D1 and Intelligent Systems. 1994. *Super Metroid*. Game [SNES]. (18 April 1994). Nintendo, Kyoto, Japan. Played August 2011.
12. Psy. 2012. Gangnam Style. Video. (15 July 2012). Retrieved August 22, 2014 from https://www.youtube.com/watch?v=9bZkp7q19f0.
13. Marilyn Schwartz. 1995. *Guidelines for Bias-Free Writing*. ERIC, Bloomington, IN, USA.
14. Statista. 2014. Number of mobile phone users worldwide from 2013 to 2019. (2014).
15. Ivan E. Sutherland. 1963. *Sketchpad, a Man-Machine Graphical Communication System*. Ph.D. Dissertation. Massachusetts Institute of Technology, Cambridge, MA.
16. Langdon Winner. 1999. *The Social Shaping of Technology* (2nd ed.). Open University Press, UK, Chapter Do artifacts have politics?, 28–40.