Behind the Scenes of Google Maps Navigation: Enabling actionable user feedback at scale

Yelena Nakhimovsky
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043 USA
yelenan@google.com

Andrew T. Miller
Google, Inc.
631 N. 34th Street
Seattle, WA 98103 USA
atmiller@google.com

Tom Dimopoulos
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043 USA
tdimop@google.com

Michael Siliski
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043 USA
msiliski@google.com

Abstract

This case study describes an Android-based feedback mechanism, created to gain structured input on prototypes of Google Maps Navigation, a mobile GPS navigation system, during real-world usage. We note the challenges faced, common to many mobile projects, and how we addressed them. We describe the user flow for submitting feedback; the resulting feedback report from the team's perspective; our triaging process for the high volume of incoming data; and the results & benefits gleaned from using this system. Learnings and recommendations are provided, to aid mobile teams who may be interested in developing a similar system for their working prototype, particularly if real-world testing is required.

Keywords

mobile research, research methods, research tools, GPS, in-vehicle navigation, in-vehicle information systems

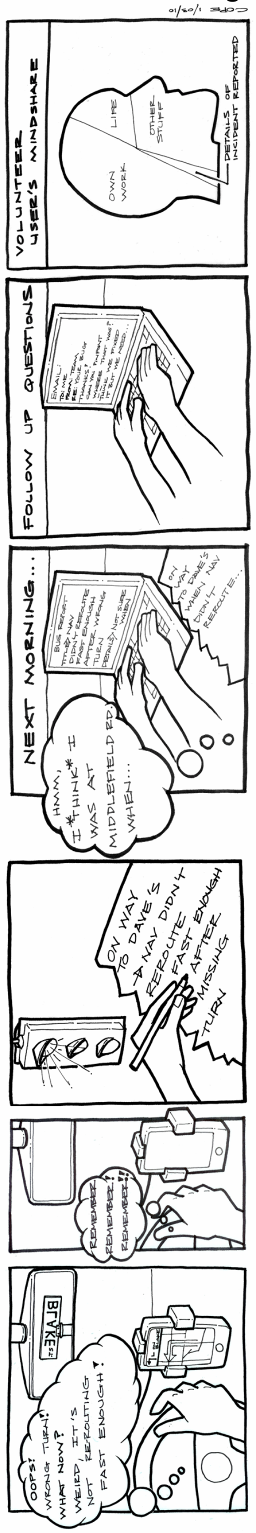
ACM Classification Keywords

H.5.2. Information interfaces and presentation (e.g., HCI): User Interfaces - Evaluation/methodology, Voice I/O, Prototyping, User-centered design

General Terms

Design, Human Factors

BEFORE the feedback mechanism


VOLUNTEER USER'S MINDSHARE
OWN WORK
LIFE
OTHER STUFF
DETAILS OF INCIDENT REPORTED
FOLLOW UP QUESTIONS
NEXT MORNING...
HMM, I *THINK* I WAS AT MIDDLEFIELD RD WHEN...
ON WAY TO DAVE'S → NAV DIDN'T REROUTE FAST ENOUGH AFTER MISSING TURN
REMEMBER, REMEMBER, REMEMBER!!
BRAKE!
OOPS! WRONG TURN! WHAT NOW? WEIRD, IT'S NOT REROUTING FAST ENOUGH!

Introduction

This poster describes a user feedback mechanism developed for use with Google Maps Navigation, a GPS navigation system meant for in-car use. The mechanism allowed for collecting actionable user feedback at scale – that is, such that it's manageable even when there's a high volume of feedback – while addressing challenges common to many mobile projects. The tool was used during internal testing, and was removed from the product before its public launch.

Aided by this feedback mechanism, during the development process we were able to identify and organize hundreds of reported issues, each accompanied by rich contextual information resulting from real-world usage. Here we discuss the challenges addressed, the design, and the results of using the feedback mechanism during development of this mobile product. This work serves as one model for researchers and practitioners to consider, as they develop any mobile product which requires user feedback in real-world settings.

Intended audience of this work

We expect this work to be of interest to mobile project teams with a working prototype, who want structured user feedback, particularly during real-world testing. In our case, the design and creation of the tool involved a user experience researcher, product manager, consumer support specialist, and software engineers.

Goals for gathering user feedback

Once the team was ready to put early versions of Google Maps Navigation into internal volunteers' hands, there emerged a clear need for handling a high volume of internal user feedback at once. There were several key goals:

- Make it easy and safe for internal users to provide feedback and bug reports during real-world testing;
- Gather timely and context-rich user feedback, in a structured manner;
- Maximize actionable feedback;
- Make it easy to triage & prioritize incoming data, at scale.

Challenges

First we had to acknowledge existing challenges, which were a combination of diary-study and in-car research issues:

- **Safety.** Safety concerns for people testing in-car devices are well-documented (e.g., [6],[7]). We determined that a simulator would not have given us the data we needed for testing, which included GPS readings, and application behavior in real-world conditions (e.g. intermittent signal loss). Thus, we needed to ensure that the internal volunteers interested in trying the application would have a simple way to give optional feedback - by pressing a unique physical button (trackball) on the device. The feedback mechanism was unobtrusive. If the user never triggered it, the UI was never presented to them. Once triggered, it was automated as much as possible (see "What we did" section for details).
- **User burden of documenting in-car feedback.** In early testing by the authors, we felt an impulse to somehow document our feedback and context. Scrawled in-situ notes proved difficult and time-

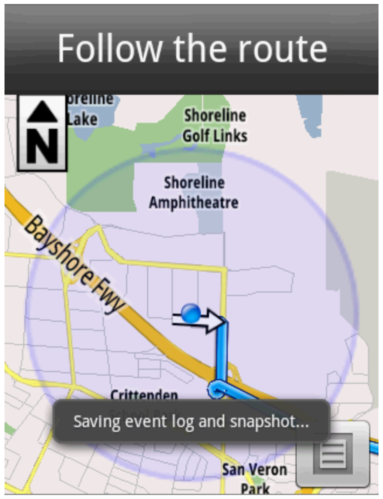


Figure 1. Prototype Navigation application, showing "Saving event log and snapshot…" tooltip, after user has triggered the feedback mechanism.

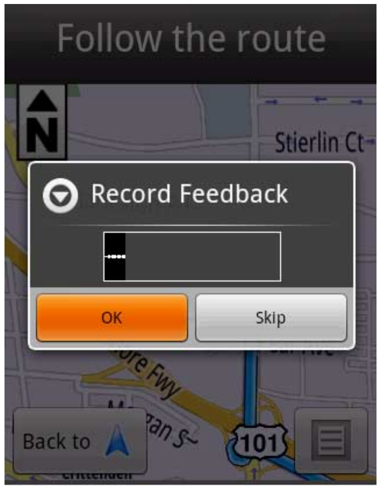


Figure 2. Fifteen second audio recording auto-starts & auto-completes, without requiring user interaction.

consuming to create, to re-interpret at a later time, and to share. Furthermore, safety considerations and the inevitable cognitive load of any note-taking motivated us to avoid this situation for users.

- **Time-delayed feedback misses critical details.** The diary study method has known challenges, including high burden for users and difficulty recapturing their experiences and context (addressed in e.g. [2][4][1][5]). Our solution made it easy for a user to capture a data-rich "experience buffer" (using an approach similar to [3]) with a button-press, within the same moment of noticing the issue, before returning to the primary task. Not only did this free the user from any perceived need to actively remember the details, but our system preserved detailed context that, in some cases, was application-centric and would have been imperceptible to the user. Furthermore, attempts to reproduce certain conditions post-hoc can be time-consuming, and sometimes fruitless.
- **Data analysis bottleneck.** In considering options for gathering user feedback, we became concerned about the potential lag time between detection of some widespread issue, and team diagnosis/response. For example, one early idea involved running a diary study employing observers and/or audio recordings; but the analysis time would not have been feasible in this case. Our triage process, described below, created clear roles and expectations for the product team members involved, and provided immediate team visibility into the data. This allowed the analysis to concurrently occur from different perspectives (e.g. some could diagnose how a system crash happened from logs, while others diagnosed a source of user confusion from a screenshot and audio clip), on a rolling basis.

What we did

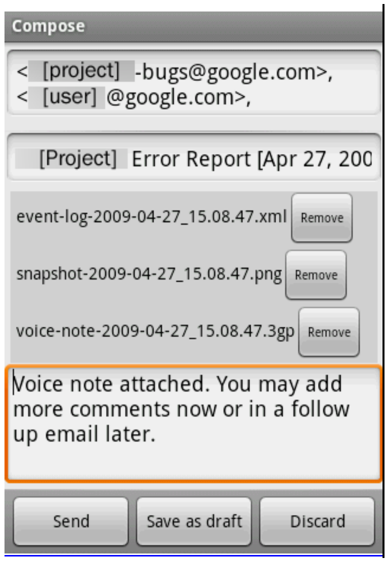
In this section we describe how we employed the feedback mechanism, how it worked from the user and product team perspectives, and how we processed the data to obtain our results.

Informed users about the feedback mechanism

We wanted to keep the feedback mechanism out of a user's way until it was needed, thus there was no explicit mention of it within the user interface (UI) of the application. The 'feature' needed to be discovered by other means. In announcements to interested internal volunteers about the availability of the app for testing, we consistently included a prominent section called "Sending Feedback," with the simple message: "Press the trackball on your Android device when you see a problem." Judging by the volume and quality of feedback received (see Results), this approach was effective.

User flow for submitting feedback

1. Driver or passenger notices a problem with the Navigation application, while it's in use.
2. User triggers feedback mechanism (in our case, by pressing the trackball).
3. Tooltip appears, "Saving event log and snapshot…" (see Figure 1).
4. 15 second audio recording auto-starts (see Figure 2). User can optionally describe the issue or context aloud. If s/he is too busy to notice/react, recording auto-completes after 15 seconds and feedback flow continues.


Compose

< [project] -bugs@google.com>,
< [user] @google.com>,

[Project] Error Report [Apr 27, 200

event-log-2009-04-27_15.08.47.xml Remove

snapshot-2009-04-27_15.08.47.png Remove

voice-note-2009-04-27_15.08.47.3gp Remove

Voice note attached. You may add more comments now or in a follow up email later.

Figure 3. Pre-populated Gmail compose window. User needs to tap "Send," but extra comments are optional. Audio clip, screenshot, and buffered event log are sent as attachments to a team list for processing, and to the user.

5. Pre-populated Gmail compose window auto-opens (see Figure 3). The user must act (i.e. tap "Send" or "Discard") to continue using the Navigation feature, but ideally this step wouldn't require user action. The email is auto-populated with:
   a. Attachments, three files: event-log (.xml), screenshot (.png), audio clip (.3gp);
   b. "To:" line: internal mailing list for bugs, and the user's internal email account;
   c. "Subject" line: [Product name] Error Report [current date & time];
   d. Body of email: "Voice note attached. You may add more comments now or in a follow up email later."
6. After email is submitted, tooltip appears, "Sending message…" to give unobtrusive feedback that the system is working.
7. User is returned to the Navigation application, which is active and running without further user action.
8. Later on, users had easy access & visibility into their own feedback via email, which proved easy & compelling to reply to, when there was additional info to convey.

Resulting feedback report, from team's perspective

The incoming data provided a more complete story of a given user-reported problem, than self-reports alone. Each feedback report was sent to a team-accessible mailing list (with a copy to the user), for our triaging process. Here is the role each attachment played:

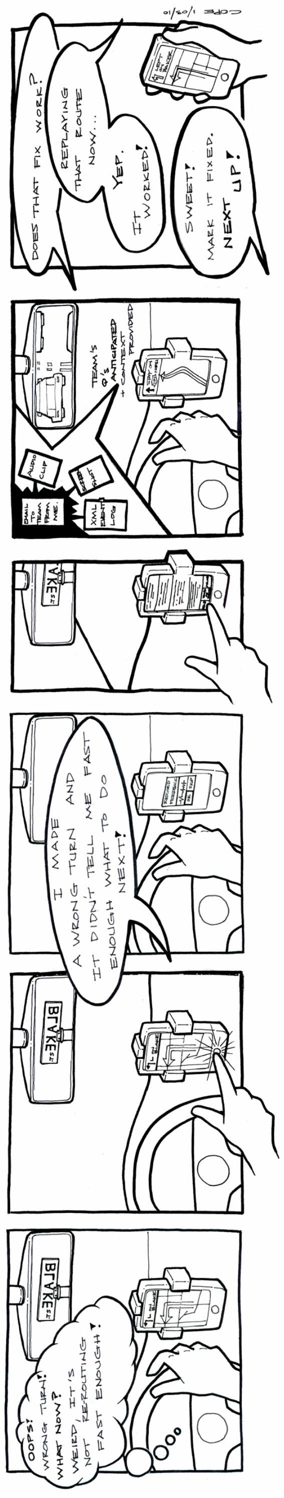
- Screenshot: Gave quick insight into issues with UI/rendering; limited burden on the Quality Assurance team to reproduce errors, when the error was visually evident.
- Audio note: Gave quick in-situ summary; tone of voice was a good indicator of user's frustration level.
- Event log: Provided user activity stream for 250 events of buffered data -- in our case, about 2-5 minutes worth of usage -- up to the point where the feedback mechanism was triggered. The destination point as well as all recent location fixes were recorded for feedback purposes. This way the route plan could be recreated, and the recent maneuvers could be replayed.
- System crash log (when applicable): Provided stack traces, giving context into the application failure, for debugging purposes.

Processing the incoming data

Based on the data received, each report was manually evaluated by a designated leader, and unresolved issues were logged in a bug database. As similar issues clustered, the severity of each could be assessed by the volume of duplicate reports. Beyond including the original attachments in the bug report:

- Audio notes were manually transcribed, with notable instances called out. Most voice notes contained relevant information and context; if clarification of comments was needed, the user was contacted for follow-up.
- Event logs were compared and cross-referenced across similar bugs, for quicker error-source identification.
- System crash logs were reviewed and commented upon in the bug report.

AFTER the feedback mechanism


OOPS! WRONG TURN! WHAT NOW? WEIRD, IT'S NOT RE-ROUTING FAST ENOUGH!
L ON BLAKE
BLAKE ST
I MADE A WRONG TURN AND IT DIDN'T TELL ME FAST ENOUGH WHAT TO DO NEXT!
RECORD FEEDBACK OK SKIP
EMAIL TO TEAM FROM ME.
AUDIO CLIP
SCREEN SHOT
XML EVENT LOG
TEAM'S Q's ANTICIPATED + CONTEXT PROVIDED
STAY ON 101 NORTH
DOES THAT FIX WORK?
REPLAYING THAT ROUTE NOW...
YEP. IT WORKED!
SWEET! MARK IT FIXED. NEXT UP!
LEFT ON BAKER
COPE 1/05/10

Reports were triaged as they were received, leading to a more rapid turnaround time for fixes & updated builds for internal release.

Results

After several months of use, approximately 1,100 feedback reports were submitted and analyzed through this system. Approximately 200 bugs were filed and prioritized.

Examples of key product improvements

- Improved timing of voice guidance;
- Stripped out superfluous instructions, lessening the user's cognitive load;
- Improved detection of whether/when to reroute;
- Improved location model accuracy, i.e. determining a user's most likely location based on imperfect GPS signals;
- Improved application performance and stability.

More traditional qualitative methods (e.g. diary studies, in-situ observations) would not have provided us with these results, particularly within the given timeframe.

Benefits of this system

- **Easier to reproduce bugs observed in the real-world.** With our Android development environment, event logs could be replayed by starting navigation with a different intent. In our case, a mock location provider replaced the system location providers, and the time-stamped location fixes in the event log could be forwarded to the rest of the application. The ability to replay an event log meant that bugs were much easier to reproduce.
- **Easier to verify bug fixes.** After a bug fix was submitted, it was easy to verify it was fixed by replaying the relevant event logs.
- **A picture is worth 1,000 words, and lots of time.** Screenshots proved to be much more helpful than text descriptions when diagnosing UI issues. Furthermore, the team didn't need to spend extra effort explaining to users how to create them, and the users weren't asked to reproduce the situation or manually create the screenshot.
- **Users remembered the details.** When the team did need clarification, after reviewing the initial feedback report, following up with users usually led to unambiguous clarifications.
- **Useful even when out-of-scope.** When the source of certain errors were outside the scope of the Google Maps Navigation application itself, problem descriptions and rich supporting data could be packaged and sent to relevant product teams.
- **More prepared for public launch.** Non-technical feedback and user behavior were invaluable in preparing the team for the types of issues we'd see upon launch, and in drafting targeted support content.

Additional learnings & recommendations

- Applying this feedback mechanism to working prototypes immediately provides a mobile product team with useful "hard evidence" as context for a given reported issue. Anecdotally, our engineering team reported it felt easier and faster to fix the issues raised.

"The new bug reporting works beautifully. Literally a joy to use."

"The feedback mechanism was brilliant and made [giving feedback] that much more frictionless and even encouraged spontaneous feature request comments.

Big kudos for putting lots of thought even into something that wasn't planned to launch!"

- Unsolicited comments from internal users

- Reports often include details that would be difficult or impossible for an end-user to perceive, or to consider relevant – particularly while driving a car.
- Triggering the feedback mechanism with an always-available action is highly effective. Ideally, the action – in our case, the trackball-press – has no other purpose within the prototype application. Otherwise, we'd suggest using "long-press" as a trigger, since the mechanism needn't be highly discoverable. (See "Informed users about feedback mechanism").
- The inclusion of screenshot and audio clip capture are valuable investments of engineering time. UI issues were reportedly easier & faster to diagnose than only text descriptions.
- User experience research becomes more scalable, when user-level data is captured and processed along with the system-level data.

Acknowledgements
We gratefully acknowledge the many internal volunteers, whose valuable feedback led to significant product improvements. Thanks also to Clint Cope, for his assistance with the before/after illustrations.

References
[1] Brandt, J., Weiss, N., and Klemmer, S. R. txt 4 l8r: lowering the burden for diary studies under mobile conditions. In *CHI '07 Extended Abstracts on Human Factors in Computing Systems* (2007). CHI '07. ACM, New York, NY, 2303-2308. http://doi.acm.org/10.1145/1240866.1240998

[2] Froehlich, J., Chen, M. Y., Consolvo, S., Harrison, B., and Landay, J. A. MyExperience: a system for *in situ* tracing and capturing of user feedback on mobile phones. In *Proc. of the 5th int'l Conf. on Mobile Systems, Applications and Services* (2007). MobiSys '07. ACM, New York, NY, 57-70. http://doi.acm.org/10.1145/1247660.1247670

[3] Hayes, G. R., Truong, K. N., Abowd, G. D., and Pering, T. Experience buffers: a socially appropriate, selective archiving tool for evidence-based care. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems* (2005). CHI '05. ACM, New York, NY, 1435-1438. http://doi.acm.org/10.1145/1056808.1056935

[4] Nakhimovsky, Y., Eckles, D., and Riegelsberger, J. Mobile user experience research: challenges, methods & tools. In *Proc. of the 27th int'l Conf. Extended Abstracts on Human Factors in Computing Systems* (2009). CHI EA '09. ACM, New York, NY, 4795-4798. http://doi.acm.org/10.1145/1520340.1520743

[5] Palen, L. and Salzman, M. Voice-mail diary studies for naturalistic data capture under mobile conditions. In *Proc. of the 2002 ACM Conf. on Computer Supported Cooperative Work* (2002). CSCW '02. ACM, New York, NY, 87-95. http://doi.acm.org/10.1145/587078.587092

[6] Rakotonirainy, A. and Steinhardt, D. In-vehicle technology functional requirements for older drivers. In *Proc. of the 1st int'l Conf. on Automotive User interfaces and interactive Vehicular Applications* (2009). AutomotiveUI '09. ACM, New York, NY, 27-33. http://doi.acm.org/10.1145/1620509.1620515

[7] Weinberg, G. and Harsham, B. Developing a low-cost driving simulator for the evaluation of in-vehicle technologies. In *Proc. of the 1st int'l Conf. on Automotive User interfaces and interactive Vehicular Applications* (2009). AutomotiveUI '09. ACM, New York, NY, 51-54. http://doi.acm.org/10.1145/1620509.1620519