Appendix to The Design of Bug Fixes

Emerson Murphy-Hill
Department of Computer Science
North Carolina State University
Raleigh, North Carolina, USA
emerson@csc.ncsu.edu

Thomas Zimmermann • Christian Bird • Nachiappan Nagappan
Microsoft Research
Redmond, Washington, USA
{tzimmer,cbird,nachin}@microsoft.com

April 15, 2013
Technical Report
MSR-TR-2013-22

Abstract — This technical report contains supplemental information to the paper “The Design of Bug Fixes”, published in the Proceedings of the 35th International Conference on Software Engineering (ICSE 2013), San Francisco, CA, USA, May 2013. The technical report contains the interview guide, list of codes used for the analysis of the interviews with example quotations, as well as the full text of a follow-up survey.

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

INTERVIEW GUIDE

I'm Emerson, a visiting professor working with RISE in Microsoft Research. I'm building analytics for how developers are fixing bugs.

I wanted to talk to you about a bug you've fixed recently.	I was just looking at Product Studio, and noticed you just resolved a bug as fixed number: _____ Is that correct? *If not, thank and leave.*

Do you have 10 or 15 minutes to chat with me?

I am here because I'm interested in **where** you fix bugs in code.

Variant 1 (for odd-numbered interviews)

There may be many potential places in code where you could fix a bug. I want to start with three examples to illustrate what I'm talking about:

- Suppose you have a UI that sometimes displayed garbled text, and it turns out a particular library call wasn't always returning what you expected. One place to fix the problem would be to use a library that does what you expect; another solution is to transform the data manually before it's shown to the user in the UI; yet another solution is to file a bug with the library authors and wait for them to fix the bug.
- Suppose you are getting a Null Reference Exception sometimes when you use the return value from method A. One place to fix the problem would be to do a null check on the return value; another place to fix it would be to change method A so that it never returns null.
- Finally, suppose you are creating a method that external clients use, but some external clients are complaining that it returns something you don't expect. One place to fix the problem is to make the original method return what these complaining clients expect; another place to fix it is to leave the method alone, but add a new method that does what these clients expect.

These are just a few examples of different places you might fix code. I'm interested in the different places you think about when are involved in a bug fix.

Variant 2 (for even-numbered interviews)

I'm going to give you a little example code; this is not a test, I'm just interested in your opinion. Suppose you have this code here, and the bug here. Take a look, and briefly sketch how you'd fix this bug.

So, how would you fix this bug?
What are alternative ways of fixing this bug?

I want to ask a few questions about the most recent bug that you were involved in that has been fixed.
Do you mind if I record audio of this?

Primary Questions

Can you tell me about the software in which you fixed the bug?

Can you briefly tell me what the symptoms were?

What was what the cause or causes of the symptoms?

Do you think that there is more than one place in code to fix this bug?

> If yes, please briefly enumerate them.
>
> If yes, why was the bug fixed where it was?

Do you feel like there was a better place to fix the bug, but it wasn't fixed there for some reason?
If so, why?

Secondary Questions

Why were you responsible for fixing this bug?

How was this bug discovered and reported?

Was this a pre-release or post-release fix?

How many versions of this software have shipped?

What phase in the development cycle is the software in? (Planning and MQ, Main Development, Stabilization, Maintenance)

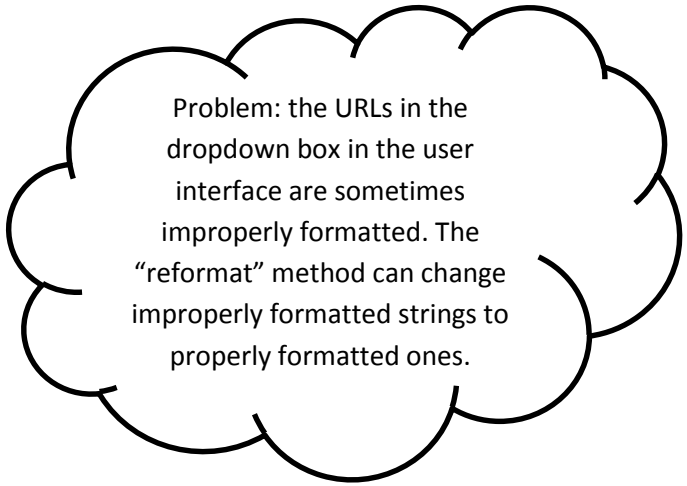
What methodology does your team use? (Agile, waterfall, somewhere between, ad-hoc, other)

Debrief

Thank you for your answers. Before I go, do you have any questions for me, or any other comments that you'd like to make?

This is the example code that was shown in the even-numbered interviews.

```
void storePeerURL(){
      …
      URL url = peer.retrieveURL();
      database.store(url.ToString());
}
…
void displayPeerURLs(){
      foreach( string url in database.entries ){
            displayInUI(url);
      }
}
void displayInUI(String s){
      urlDropdown.add(s);
      urlDropdown.setEnabled(true);
}
…
string reformat(string oldUrl){/*an predefined method*/}
```



CODE LIST (ANALYSIS OF INTERVIEWS)

We used the following codes for the analysis of interviews (in alphabetical order).

alt fix

analytics

application

collaboration

dependents

feature request

methodology

people with similar problems

phase

real fix

related changes

release

roll back

symptom

thought just one fix

version

why not alt

EXAMPLES QUOTATIONS FOR THE CODES (ANALYSIS OF INTERVIEWS)

This section presents for each code examples of quotations from the interviews.

Code: alt fix

if I have all the time in the world, I would make sure I have procedures to mine the ranking features (P6)

the other option is not to fix it and not to investigate it. (P7)

The longer-term fix would be to choose a different architecture (P11)

Put a big old catch everything on it and problem solved (P15)

try to parse the query, understand how many rows should be affected, for example, and verify just the original table without having this clone. (P24)

Code: analytics

Do I really want to go quantify this and say - you know, go through all these things and say, “Hey, you know, for every test, we're really wasting five minutes per test”? (P10)

A feature that I’m not really sure if anybody actually cares about. (P34)

Code: application

Backend service. (P1)

alerting system (P8)

desktop app. (P11)

Code: collaboration

I sent the proposal to the dean, it got reviewed, and discussion and finalized (P22)

I also have to talk to one of the other developers (P28)

we exchanged e-mails and sort of discuss about like what would be the best way to do (P34)

consensus from the DM, my dev lead, and the person who found that recording, and everybody was okay that that’s the thing. (P34)

noticed the bug in code review (P35)

Code: dependents

many users - users that are using the system. (P2)

Code: feature request

it might be case specific it's a bit difficult to, I think, keep track of. (P17)

The testing team filed another bug and they filed it because they were going to consider fixing it in another milestone. (P25)

Code: methodology

somewhere in between agile and ad hoc. (P1)

Waterfall (P11)

Seven to eight. [As response to a follow-up question “Agile? Okay. Like on a scale from one is waterfall. Ten is agile. Where would you say you are?”] (P13)

Code: people with similar problems

It seems like it would be nice to - it would still be nice to have it documented somehow, so, even though it is low, at some point, like when we’re deciding what big changes we want to make or - at least other people, who have sort of similar problems, if it’s just you - right - maybe it’s not a huge priority. (P1)

Code: phase

Maintenance (P3)

Stabilization (P37)

Code: real fix

easy fix is to just resubmit the job (P3)

It was in their code. Okay, so you decided that they're the ones that should fix it. (P16)

make sure that the focus doesn’t go away from editable area. (P37)

Code: related changes

not only did you fix the bug at where the deadlock was occurring, but some other possible places where deadlock could occur in the future (P13)

Code: release

Pre-release

Post-release

Code: roll back

But then I don't know if we ever go back and kind of “Oh, okay, we had to do this, now we can change it.” (P15)

Code: symptom

Every page refresh or navigation was chewing up about a meg of memory (P7)

There was an error message. It said the compiler crashed. (P9)

I think it was actually an assertion failure that a certain variable was zero and was not expected to be. (P9)

We had a button in the UI that wasn't doing what it was supposed to do. (P11)

Test that failed at some point (P14)

Code: thought just one fix

I think it was the optimal way. (P8)

[…] the fix was sort of the only possible way to fix the problem, and therefore, it was the best fix. (P31)

Code: version
[This code was used for responses to question “How many versions of software have shipped?”]
This is our first version

More than ten

Two or three

I don’t know.

Code: why not alt

I don’t have access (P3)

Take you awhile to find the right person (P3)

People who are responsible for it were busy doing something else (P9)

Basically that's a huge change. (P11)

There's a risk that something else breaks, right. (P15)

I mean we worried about not to introduce a lot of bugs without commit. (P22)

It was simpler than, you know, I didn't want to complicate my code with all this new statements. (P23)

Obviously you also have to think about where we are in the project cycle (P30)

Do they ever actually call this function with [a specific flag]. So in that way, it’s not the best fix. (P34)

I guess the return on investment wasn’t worth it for a test aspect. (P36)

SURVEY

This is the full text of the survey that was sent to employees.

MS Research Survey on Bug Fixes

We are researchers in the Empirical Software Engineering (ESE) group at Microsoft Research who are interested in understanding bug fixing and building recommendations for better bug fixing practices and tool support. We would appreciate your feedback via this 20-minute survey, since it can help us refine our hypotheses and develop better recommendations. For more information, please contact Emerson Murphy-Hill, Tom Zimmermann, Chris Bird, Nachi Nagappan. As a thank you for your time, you can enter your name into a raffle of two $50 Amazon gift certificates at the end of the survey.

This survey is anonymous

Demographics

1. What division do you primarily work in?

   - Business Solutions
   - Interactive Entertainment
   - Office Division
   - Online Services
   - Server & Tools
   - Skype
   - Windows & Windows Live
   - Windows Phones
   - Other

2. Which best describes your primary work area?

   - Development
   - Test
   - PM
   - Build
   - Design and UX
   - Documentation
   - Other

3. How many years have you worked at Microsoft? (decimals okay)
(Min Number: 0 - Max Number: 40)

4. How many years have you worked in the software industry? (decimals okay)
(Min Number: 0 - Max Number: 40)

5. What percentage of your immediate team works in the same office/location as you?
(Min Number: 0 - Max Number: 100)

Bug Fixes

Some bugs can be fixed in multiple ways. For example:

- A bug that propagates bad data through several layers may be fixed in any layer between the source and the user interface.
- A bug may be fixed by preventing an exception from being thrown, or by catching that exception before the user sees it.
- A bug may fixed directly or may be refactored first and then fixed.

6. Of the bugs that you fix, approximately what percentage are there multiple potential fixes?
(Min Number: 0 - Max Number: 100)

In the remainder of this survey, we will be asking about only those bugs for which there are multiple fixes.

7. Of these bugs you fix, which of the following is most common?

- No fix will satisfy all stakeholders.
- Only one fix will satisfy all stakeholders.
- More than one fix will satisfy all stakeholders.

8. How often do you personally feel satisfied with the fix you applied?

- Never
- Rarely
- Sometimes
- Usually
- Always
- N/A

9. Once you've found the bug and there are multiple potential fixes, how often is the appropriate fix immediately apparent?

(As opposed to you needing to carefully consider the benefits and drawbacks of each potential fix.)

- Never
- Rarely
- Sometimes
- Usually
- Always

- N/A

10. How often do the following factors influence which fix you choose?

	Never	Rarely	Sometimes	Usually	Always	N/A
Changes few lines of code	○	○	○	○	○	○
Requires little testing effort	○	○	○	○	○	○
Takes little time to implement	○	○	○	○	○	○
Deployment cost is low	○	○	○	○	○	○
Creates code that will be easy for future developers to understand	○	○	○	○	○	○
Doesn't change external interfaces or breaks backward compatibility	○	○	○	○	○	○
Maintains the integrity of the original design	○	○	○	○	○	○
Meets quality standards	○	○	○	○	○	○
Phase of the release cycle my product in	○	○	○	○	○	○
Whether users are going to find it intuitive	○	○	○	○	○	○
Whether I'm the most qualified person to implement this fix	○	○	○	○	○	○
My peers' opinions of the fix	○	○	○	○	○	○
My manager's opinion of the fix	○	○	○	○	○	○
There's a standard fix for this particular type of bug	○	○	○	○	○	○

11. If there are any other factors that influence what fix you choose, please enter them here.
(Max Characters: 2000)

12. How often do you apply the "optimal" fix for a bug?

An optimal fix is the one you would implement if you were not limited by time.

- Never
- Rarely
- Sometimes
- Usually
- Always
- N/A

13. For those bugs for which you did not apply the optimal fix, which are the following are reasons for you not doing so?

(Check at least 1 but no more than 11)

- ☐ I didn't have time to figure out why the bug occurred
- ☐ I didn't have time to implement the fix
- ☐ I didn't have time to completely understand the APIs I'd have to use
- ☐ I didn't know where to find the necessary code
- ☐ I didn't have permission to modify the code

- ☐ The software wasn't important enough
- ☐ The documentation of the code was poor
- ☐ The bug appeared so rarely that a non-optimal fix will suffice
- ☐ I couldn't find the right person to help me make the fix
- ☐ The right person to help me make the fix had more important things to do
- ☐ Other (specify below)

14. If you have reasons for not implementing optimal fixes that are not listed in the previous question, please list them here.
(Max Characters: 2000)

Reconsidering Fixed Bugs

15. For bugs that you fix sub-optimally, how often do you think an optimal fix should be reconsidered in the future?

- ○ Never
- ○ Rarely
- ○ Sometimes
- ○ Usually
- ○ Always
- ○ N/A

16. What is the most common mechanism you use to make sure that the optimal fix is considered in the future?

- ○ Make a mental note
- ○ Make a comment in the code (e.g. TODO)
- ○ Write a comment on a bug report
- ○ Create a new bug report
- ○ Keep a private list of the bugs I'd like to see fixed in the future
- ○ None of the Above

17. What are the advantages and disadvantages of using the mechanism that you selected above?

If the mechanism you use is not listed, what do you do?
(Max Characters: 2000)

18. For bugs that you do not initially fix optimally, how often do they get fixed optimally later?

- ○ Never
- ○ Rarely
- ○ Sometimes
- ○ Usually
- ○ Always

- N/A

Bug Fixes and Software Usage

19. How often is your choice of fix dictated by how often the situation in which the bug was reported actually occurs in practice?

For example, if the bug is that .mdb files are deleted when updates are applied over a period that includes midnight, which fix you apply may depend on if users frequently have .mdb files and also install updates at midnight.

- Never
- Rarely
- Sometimes
- Usually
- Always

20. What is the most common mechanism you use to determine the frequency of such situations?

- Guess
- Estimate based on my past experience as a user of the software I develop
- Estimate based on my past experience interacting with users
- Collect data by taking a quick convenience sample (e.g., ask devs on my team)
- Collect data by external polling (e.g., ask readers of my blog)
- Estimate based on existing usage data that I remember seeing in the past (e.g. SQM)
- Write a query over existing usage data (e.g. SQM)
- None of the Above

21. What are the advantages and disadvantages of using the mechanism that you selected above?

If you commonly use a mechanism not listed above, what is it?
(Max Characters: 2000)

Collaboration

22. When choosing which fix to apply to your bugs, how often does the decision get made in the following ways?

	Never	Rarely	Sometimes	Usually	Always	N/A
I choose the fix.	○	○	○	○	○	○
My manager chooses the fix.	○	○	○	○	○	○
My team collectively chooses the fix.	○	○	○	○	○	○

23. How often does communicating with the following people help you choose the optimal fix?

	Never	Rarely	Sometimes	Usually	Always	N/A
Peer SDEs	○	○	○	○	○	○
Peer SDETs	○	○	○	○	○	○
My manager	○	○	○	○	○	○
My product manager	○	○	○	○	○	○
The people who wrote the code related to where the fix might be applied	○	○	○	○	○	○
Other experts (e.g., architects)	○	○	○	○	○	○

24. When discussing which fix to apply with other people, by what means do you most often communicate?

- ○ In the bug report
- ○ By email
- ○ Unplanned meetings
- ○ Planned meetings
- ○ None of the Above

25. If your answer was "None of the Above" to the previous question, how do you communicate? (Max Characters: 2000)

26. For bugs you fix, including yourself, how many people are typically involved in ...

	1	2	3-5	6-10	11+
finding the cause of a bug?	○	○	○	○	○
choosing a solution?	○	○	○	○	○
implementing the solution?	○	○	○	○	○

Other Questions

27. How often do you choose not to file a bug report for the following reasons?

	Never	Rarely	Sometimes	Usually	Always	N/A
I don't know where to file the bug or who to report it to	○	○	○	○	○	○
The bug is unlikely to ever be fixed	○	○	○	○	○	○
Adding another report makes it look like the software is of poor quality or that the team is behind	○	○	○	○	○	○
Whether or not the bug gets fixed has little impact on the software I'm developing	○	○	○	○	○	○
A bug puts pressure on a colleague to fix the problem; I don't want to add to his or her workload	○	○	○	○	○	○
Filing this bug dilutes the urgency of bugs I think are more important to fix	○	○	○	○	○	○

28. How often do you remove or disable the feature that a bug is in, rather than fix the bug itself?

- Never
- Rarely
- Sometimes
- Usually
- Always
- N/A

29. How often do the following situations occur when you are fixing a bug:

	Never	Rarely	Sometimes	Usually	Always	N/A
I notice code that should be refactored.	○	○	○	○	○	○
I refactor this code that should be refactored.	○	○	○	○	○	○

30. If you do not always refactor code that should be refactored, why not?
(Max Characters: 2000)

31. If you've worked on a project that has had a "bug cap," have you taken any actions to artificially reduce the number of bugs that you were assigned?

If yes, what were those actions?
(Max Characters: 2000)

32. In your opinion, what's the biggest challenge developers face when fixing bugs?
(Max Characters: 2000)

33. If you have any other comments on bug fixing or this survey, please enter them here:
(Max Characters: 2000)

Thank you for your feedback. Before you submit this survey, please click this link to send an email and enter in a drawing for one of two $50 Amazon.com gift certificates (this keeps your responses anonymous).

