Wildbit

The Blog

Thoughts on building web apps, businesses, and virtual teams.

Twitter

...

11 Nov The importance of commit messages ← Go back

Posted by Petyo Ivanov on November 11, 2008 — 2 Comments

> Once you get past the prototyping, unit tests and the code what you have is the only real documentation anyone is going to update.

— Something that sounds like "Code Complete" or another McConnell classic.

> A good software development process is all about communication.

— 37 Signals about Basecamp.

> Thank you, Captain Obvious

— Anonymous lolcat.

Communication is the mantra of two of my long-time feeds. Proper communication of your actions adds tremendous value to your work. Without good communication skills you're nothing but a lone 'hacker' doing stuff for yourself. And what can be better than talking with the team without leaving your favorite text editor/IDE?

Not long ago I was still considering version control systems as offsite backup for my code and (at best) a protection tool when developing software in a team. From time to time I was closing bugs using "Fixes #145" in my commit message, without being very diligent in it.

After joining Wildbit, one of the first things that really astounded me was how the Beanstalk dashboard looked for our projects. Insightful commit messages, references to solved problems; task status reports. I could easily see what the entire team was doing. “Looks awesome” I thought. “Why I haven’t seen this before?” I thought. Needed some time to fully grasp the concept – when you’re a virtual, **the commit messages are (one of) your primary means of communication with the rest of the team.**

It did not take me much time to get addicted, and to start “stalking” the guys’ work via RSS (I love picking the brains of smart people, and we have plenty of them here). Reviewing the daily commits of my team mates gave me instant programming skills boost. Usually, I read feeds through Google reader, but for our source code feeds I use Liferea* – for real time awesomeness.

* The only one in the team using Ubuntu for daily work. Most of the guys pledge loyalty to apple products.

Let’s do like the cool kids, but how?

Trust me, you don’t want to be the dunce, who commits with blank messages, or with meaningless ones like “Fixes”. Your team will hate you. You will hate yourself after a month, when trying to discover what the hell this code does in your brillant project. So, let’s start paying attention to this, shall we? As time passed I discovered that

thinking in terms of your next commit helps you stay focused on the task at hand.

Do one thing at a time. Sounds easy, but I frequently fall in cases when I have to break down my work into several commits for the sake of clarity. This is a Bad Thing actually – a sign that I did not stay focused on a task, but instead did several non-related changes. And this was most visible when I was prompted for commit message – I was having hard time describing what I did. Did this mean that I did not do the right thing? After a while, I tried to define the commit message first, then start working on the task. The commit message is like the goal for your coding session.

How to write good commit messages – the definitive guide.*

* Yeah, I wish.
The code should stay connected to the specs. Commit messages are the only means of doing it. Refer to the specs, whenever possible.

Docs and plans become outdated at the moment stuff described in them is implemented, and you discover that things did not happen like you thought. By linking your commits to specific document, you breathe life to the specs, providing additional information about how it goes, what *actually* happened, what changed, while staying DRY.

Besides, if you don't describe your actions in the moment of commit, you're doomed to repeat what you did again and again in meetings, tickets, reports.

Most of the modern bug-tracking and project management systems provide integration with the source control repository, but even if they don't, you can still drop task/ticket numbers in the commit message – something like this. Of course – don't rely on this being the only description of your message.

```
See #568. Closes #102. Attempts to resolve #12.
```

Mind that the commit messages may be reviewed with or without the code.

This is actually a tricky part. The commit message should make sense without seeing the actual code, but still should not repeat the code changes. The best way to fully grasp this is to review the log of your project, preferably commits from someone else. What makes sense? What does not? Why? Through time, you will get used to proof-read yours.

Fix:; New:; Enhancement:; Sorry:; – what did you do?

Since the days of Bugzilla, Inserting formal notations in the commit message is not something new. By distinguishing the different types of action you do on the project is really helpful. Later, you can filter out only the ones who were bug related, or your sorries. Though the types are self explanatory, I will restate them.

- Fix: – closing bug.
- New: – implemented something new.
- Enhancement: – we're making it better.
- Oops: – Even the best make mistakes, commits containing typos or plain non-working code. Show your shame!

Feel free to determine what works for you and your team.

Reveal your intentions, don't describe what the code does. Tell why it does it.

This is actually a statement you have already heard a thousand times, regarding the comments in the code. I avoid comments like plague (but this is another topic). However, this statement can be fully applied to the commit messages too. Just keep your mind on the purpose, not the implementation.

Leave insights, skip profanity.

I frequently discover new stuff about the programming languages I use. Neat and natural ways of using inject, for example. Don't hesitate to drop a line or two for your findings – this may help others understand what actually happens. Sometimes, I am hesitant about the result of my session. In such cases comments like "Hum, this does not look right; Ilya, please see if you have the time." may turn your reviewers' attention to a problem.

One last thing – don't swear. I know. He does it. But you should not. No matter how angry are you. Feel free to swear in the traffic jam, but keep your code clean. I used to swear in comments, but do not do it anymore. You never know who will encounter it. And it will definitely be a plus for you.

Stay concise, be up to the point.

And after all – don't become too verbose in your prose. A long commit message is usually a bad sign for too much stuff done in a single commit. A good commit message is usually one or two sentences.

Finally, I want to share some good and some bad commit messages from one of our latest projects. The bad ones are usually courtesy of yours truly.

The good:

- remove suspended videos from playlist
- Recover canceled beta subscription added.
- Save ReleaseServers without validations in migration script.
- Getting rid of stupid schema.rb that don't understand what BIGINT is. Use SQL dump instead.
- Whoops! We need our server_definition back here.

And, some bad:

- correct config — *Why? what was wrong before?*
- Production related fix scripts — *Actually, this is one bulk commit of totally unrelated stuff*
- Another unexpected error — *Does not make sense without the code.*
- error container — *What? Have you heard about punctuation?*
- tag fixes — *No meaning; might as well be left blank.*
- lets not spam anyone. — *Nice one. Anything else? Wishes for world peace would work too?*

2 Comments

2 Trackbacks/Pingbacks

1. Pingback: Freelancer Pro-tip: Be on the Communication Offensive on November 19, 2008
2. Pingback: A Leased Line to the Collective Unconscious » Subversion for web design - how we do it @ Karyx on December 3, 2008

Write a comment

Name * required

Email * required

Website

Your comment, no HTML please...

Post comment

Subscribe to our feed

Hi, we're Wildbit.

We create web products such as Beanstalk and Newsberry to help business collaborate and communicate more effectively. Meet our team, view our client work or get in touch!

Our Products:

- Beanstalk
- Newsberry

← Go back

Get in Touch

Wildbit, LLC

Work 20 North 3rd St, 2nd Floor
Philadelphia, PA 19106 USA

Work Phone
    +1 (215) 203 0488

Fax
    +1 (267) 200 0835

Email
    info@wildbit.com

We work at IndyHall. Coworking is more than just space.