ArticleS. TimOttinger.

ApologizeIncode [add child]

Apologies In Code

I was thinking about the whole "bad comment smell" the other week when teaching a class. Well, actually I was talking out loud about it, and the whole class got mad at me. They felt that comments were very important and should be included. They put all kinds of information into comments, information that would otherwise be part of the version control log, mostly. There were explanatory comments all over the place. A week or two later, I was reviewing coding standards that demand comments for many purposes and in many places. I then attended a code review where the analyst called for more comments.

Bah, humbug.

A comment is an apology.

A comment is an apology for not choosing a more clear name, or a more reasonable set of parameters, or for the failure to use explanatory variables and explanatory functions. Apologies for making the code unmaintainable, apologies for not using well-known algorithms, apologies for writing 'clever' code, apologies for not having a good version control system, apologies for not having finished the job of writing the code, or for leaving vulnerabilities or flaws in the code, apologies for hand-optimizing C code in ugly ways. And documentation comments are no better. In fact, I have my doubts about docstrings.

If something is hard to understand or inobvious, then someone *ought* to apologize for not fixing it. That's the worst kind of coding misstep. It's okay if I don't really get how something works so long as I know how to use it, and it really does work. But if it's too easy to misuse, you had better start writing. And if you write more comment than code, it serves you right. This stuff is supposed to be useful and maintainable, you know?

Is there any use of comments that are not apologies? I don't think so. I can't think of one. Is there any good reason to write a comment? Only if you've done something "wrong".

And that's all I have to say about that -- F. Gump
PS I found that this was all discussed a long time ago on Ward's Wiki. It's not just me. -- Tim

!commentForm

Wed, 28 Jun 2006 10:57:43, Eber Irigoyen, some examples *Expand All* | *Collapse All*

I use comments for a lot of things, not only to describe what a function does or certain parts of the code, but you are assuming that your code is readable, that everyone would've done it the same way you did it, that everyone can understand your code, that everyone will have access to the version control (instead of just the code), etc...
comments can also be used to quickly describe critical parts of the program, such as technologies/components used in the project, names and dates of creation, changes, version of software used to develop, etc

http://ebersys.blogspot.com/2006/04/bare-minimum-required-comments-for.html

Wed, 28 Jun 2006 11:23:00, Tim Ottinger, Misplaced? *Expand All* | *Collapse All*

Yes, everyone should be able to understand your code. It was written to programmers, not just for compilers. That is what high-level languages are for.

And it should have tests which tell us what the programmer considered when writing it.

If someone would have done it differently, they still should be able to tell how I did it, if I'm a reasonably good programmer.

Technologies and components need to be documented elsewhere, not where they're being used. And they should be obvious. You shouldn't have to apologize for using them.

Version control has names and dates of creation, and better information besides. Version of software used shouldn't matter.

Generated code should be commented, because we should have to apologize for using code generators.

The code is an artifact, and I don't care if you used bear skins and flint knives to write it. The product should be the product. If you don't have the version control, then maybe someone didn't want you to know when/who. And when/who just tells you who to complain to -- it's an apology in advance: "if you don't get it, tell frank, who wrote it back in 77 so it isn't very good or modern."

I think all those uses are apologies.

Wed, 28 Jun 2006 11:30:52, Stan James, Why? *Expand All* | *Collapse All*

Some comments tell why things were done. Here's a real one: "This looping algorithm replaces the obvious recursive solution which blew out the stack".

Wed, 28 Jun 2006 12:46:00, Paul Pagel, Are todo's acceptable *Expand All* | *Collapse All*

//TODO: <SomeDevelopersName[?]>: This is very ticklish. If you clone the <SomeDomainObject[?]>, it fails.
// Something about our little world has broken basic polymorphism (dynamic dispatch).

I came across this yesterday.

What about TODO's which sit and rot in the code for ages. Does anybody actual search them out? In my experience they become stale and a part of the system. Should they be left out? Really if a todo is needed, it should be finished in the scope of the story, right? So temporary todos could be a justified comment if there is developer discipline to go and do the TODO. I personally think little notes on a notepad work better, cause i don't always remember where to look.

Wed, 28 Jun 2006 14:36:31, Tim Ottinger, Why Expand All | Collapse All

```
Some comments tell why things were done. Here's a real one: "This looping algorithm
replaces the obvious recursive solution which blew out the stack".
```

Apologizing for the fix or the bug? Not sure.

I think that users of the program don't necessarily need to know (two, five, or one hundred) iterations from now that there used to be a bug. If they did, the most code would be 98% comments. What you're quoting is a good check-in comment in version control.

Wed, 28 Jun 2006 15:55:27, Paul Pagel, Expand All | Collapse All

I agree...If there is a complicated algorithm which needs to be explained better, break it up into methods and clear variables which can do the explaining. Dont use int i = 0; Make it something meaningful.

Wed, 28 Jun 2006 16:32:32, , Presumption of guilt Expand All | Collapse All

Sometimes you have to apologize, but it's still an apology. I think that there should be a presumption of guilt though. It's depressingly common to code review stuff that relies on comments to cover for poor quality - this is a sad reality. It's certainly true that frequent inline comments have a very high correlation with multiple responsiblity methods. Commentless code is a high standard - perhaps too demanding, but combined with a focus on clear intent in a code review it's an excellent goal.

Wed, 28 Jun 2006 16:33:59, Anthony Bailey, Higher-level concepts Expand All | Collapse All

At the level of lines of code and method declarations, I concur that a comment is an apology. I think a good comment is a good apology - one that shows the apologizer knows what they've done wrong. Comments often contain the seeds of their own destruction - you can translate them into directions as to what to change to make them unnecessary. So at least comments are the kind of smell that comes with instructions for deodorizing.

Comments at the level of classes and packages (or similar) I have more time for. Perhaps they are partially apologies for not having good enough navigation in development environments or representative powers in the languages we use. But perhaps

comments are in essence more necessary for these higher-level concepts. In natural language writing, higher-level concepts and abstractions often need a definition or an introduction; the concept is genuinely complicated enough to need a sentence to express it, and then we introduce a more succinct name for the concept for ease of reference.

Wed, 28 Jun 2006 18:31:03, , Class Comments and Interface Comments Expand All | Collapse All

The only kind of comments I like are class comments (before a class, saying what's the purpose of the class) and interface comments (javadoc comments before each method in an interface, so that the IDE can display what each parameter is used for).

Other that that, comments are apologies, I agree, and maybe they should be banned from production code.

Wed, 28 Jun 2006 23:04:15, NaimRu[?], Baby with the bathwater? Expand All | Collapse All

Sure, comments should never replace refactoring for clarity. And as a training technique, you may want to ban some folks from adding comments to assist in breaking bad habits. But just because common practices for commenting code are "smelly" does not mean that all comments are harmful. I know some in the "pure XP" community may disagree, however, there are many times when I feel it is quite appropriate and helpful to have this kind of metadata about the code right there in the code. A good example is a comment that answers "why" a choice was made -- e.g. the selection of a particular algorithm over another.

But what about the XP maxim that comments can lie, but code always tells the truth? Comments are a form of documentation. XP preaches the elimination of ***unneeded*** work. Anything you create must be maintained in synch with the code (so it does not become a lie) or it should be conciously discarded. However, this does ***not*** mean the elimination of all documentation (including comments). IMHO, XP gets a little bit of an undeserved black eye when this is not stated carefully.

Wed, 28 Jun 2006 23:34:31, NaimRu[?], Code generation considered harmful? What?? Expand All | Collapse All

```
Generated code should be commented, because we should have to apologize for using code generators.
```

I thought all compilers were code generators. Code generation is just a way of taking ideas expressed in a higher level language and automatically translating them into a lower level language. Why should we apologize for using code generation as means of encoding our abstractions? It OO the "one true way"? But I digress. :-)

Thu, 29 Jun 2006 04:22:15, John Wilkinson, Why? Expand All | Collapse All

// // Some comments tell why thing are done...

// Apologizing for the fix or the bug? Not sure.

// I think that users of the program don't necessarily need to know (two, five, or one hundred) iterations

// from now that there used to be a bug. If they did, the most code would be 98% comments. What you're
// quoting is a good check-in comment in version control.

Such a comment will deter a programmer from replacing the complex looping algorithm with the simpler but inappropriate recursive algorithm. Even if there is a test, as there should be, to catch the reappearance of the stack overflow bug, the time take to replace the algorithm will have been wasted time.

▼ *Thu, 29 Jun 2006 11:01:37, Stéphane Tavera, In tests ?* Expand All | Collapse All

I was thinking to ask you the following question :
Do you relax your rule in unit tests ?
I am very often inclined in commenting inside a unit test to express what I want to test.
Your article lead me to apply this rule also in unit tests by spliting some tests in more specialized ones with a meaningfull name.
No more need to comment.
So, thanks for your article !

▼ *Thu, 29 Jun 2006 13:09:57, Stan James, Comments explain why ...* Expand All | Collapse All

"I think that users of the program don't necessarily need to know (two, five, or one hundred) iterations from now that there used to be a bug."

What I wanted them to know is when they see the looping code and say "Hey, that would be much better [however you define better] with recursion!" they shouldn't jump in and "improve" it. It might even be worth avoiding a puzzled look at a few minutes wasted figuring out why such an obvious design was not used.

▼ *Thu, 29 Jun 2006 13:25:54, Tim Ottinger, Baby with the bathwater?* Expand All | Collapse All

You know, I used to preach commenting a lot. There might even be the odd person (very odd) who remembers when I used to push for 60% commenting (by LOC) in C. I used to push for C++ class comments, and function header comments. Even flowerboxes. When I went to python, I immediately fell in love with the docstrings. I thought doxygen was a cool idea.

I still have some lingering feelings of fondness for python docstrings, but now I don't know that I need them. When they're interesting and useful is when they are examples of usage (not description, but actual examples embedded in the string). But now I also have unit tests. Sometimes (when they're well-written) they are better than the examples, but I don't always know how to find them in other people's code.

Maybe examples in the docstrings are apologies for not having really good access to the test examples. I don't know. But I need comments less and less, especially other people's comments. I just don't need them like I used to. When I do need them, I can't help asking myself why I needed them. That makes me want to fix things.

Years and years ago I inherited some code that was very liberally documented. After struggling with it for a day or so, I ended up using awk to strip all comments from the code, and then I started renaming things. The next day I was done with my changes, and everyone was expecting me to take a week or so. That was back in C days, not in modern IDE and modern language days. There was no XP, and that experience has stuck with me.

I find myself feeling guilty when I write a comment now, and I start examining my motives. What is it that is inobvious? Why do we need it? What have I done wrong? You get the sense of it.

I'm thinking that we don't want to throw out the baby with the bathwater, but we don't want the baby to live his life in the bathwater either. There comes a time to separate the two.

Thu, 29 Jun 2006 19:20:26, Tim Ottinger, Comments explain why ... Expand All | Collapse All

I would be willing to concede that comments to save processing time are a good idea but I think that kind of comment is solving a problem that doesn't really exist (or close enough to "never" to be statistically insignificant).

On a software development team, people generally don't go around looking for code to rewrite simply out of a sense of adventure or duty or aesthetics. Typically, people don't want to change code, unless there is some requirement (bug or otherwise) that forces them to make some change. If that code works, then the fact that it used to not work isn't very useful. As long as it works, it probably won't be modified at all. It makes sloppy version control unit-of-work.

If it needs to be changed, then maybe it's not so bad to have them try to make a better recursive function to do it. Clearly the old one won't work, but maybe a better recursive algo would be okay. I guess it's a matter of stack size and available stack space and when/why you recurse. MOst of my recursive stack overflows have been an indication that my algo wasn't smart enough, so I might be projecting here. But I fixed a horribly naive recursive algorithm last month with a moderately smart recursive algorithm and it no longer blows its stack and has many orders-of-magnitude better performance.

If there is a reasonably good test case or simulation, then maybe someone will ignore the comment and try it and the code might be the better for it. If it doesn't work, then they'll know it doesn't work, and maybe they'll write a better test case to prove that it won't work that way.

But what do I know?

Fri, 30 Jun 2006 10:15:31, Pierre, Comment can be documentation Expand All | Collapse All

My mothertongue is French, but I code in technical English for colleagues who are not speaking French, so I like to add comments not as an apology for my "poor technical English" but as a precision for colleagues who speak another language, and might not understand English.
I also like to find "how to invoke" comments in an API, especially when they appear in a yellow box like in Visual Studio: handier than having to push F1, lost focus etc.
I like explanations about why a particular path was chosen among several possibilities.
I also use to first write my code as comments (scaffolding), why should I remove them afterwards?

Sat, 8 Jul 2006 17:00:16, Ed Kirwan, Dogfood Expand All | Collapse All

Hi, Tim,

I apologise in advance for being a not-too-often visitor to these blogs, so the answer to my request may already be out there in

the Object Mentor website, but I have to ask anyway.

Most times, code-snippets are fine for showing how a certain thing can/should be done; but your position here is so antagonistic that you really have to eat your dogfood.

Show us some code you've written.

Not a snippet, not a slice, not a method: an application.

Show us even a 1000-line, uncommented applicationlette. Don't tell us anything about it. If we can see precisely what all the parts do, then you win.

.ed

PS I love dogfood; you can see an 8000-line (15,000 if you include, "Apologies") application I wrote at www.edmundkirwan.com

PPS I also admit that we don't all have the luxury of time: you may just be too busy to write open-source code. I fully understand this.

Sun, 9 Jul 2006 18:01:05, Ed Kirwan, Joggin Expand All | Collapse All

Actually, I went jogging today and I thought of something else.

I presume your stance is: "Comments should not be written." This is because the variables and method-names should be self-evident.

Well, to play devil's advocate, here's another one for you: automated unit tests should not be written. This is because the code should work.

Sun, 9 Jul 2006 23:56:44, Uncle Bob, It should just work. Expand All | Collapse All

> Ed Kirwan said: *"automated unit tests should not be written. This is because the code should work."*

It's not quite that extreme. Some comments are worth writing. Many, however, are excuses for not writing clearer code. Often we look at the code we just wrote and say "Wow, that's messy! I'd better comment it." Instead we need to be cleaning that code up.

I will write comments; but when I do I consider it to be a failure of expression. Sometimes the fault is our languages. Sometimes the fault is our own talent or imagination.

Mon, 10 Jul 2006 04:52:29, Ed Kirwan, Uncle Bob's clarification Expand All | Collapse All

Uncle Bob wrote, "Some comments are worth writing. Many, however, are excuses for not writing clearer code. Often we look at the code we just wrote and say "Wow, that's messy! I'd better comment it." Instead we need to be cleaning that code up."

That's much more palatable.

(Now, where's that Roman Numeral converter again ...)

.ed

Mon, 10 Jul 2006 18:53:29, Tim Ottinger, put up Expand All | Collapse All

You're right. I should put up some code. Also that I have been too busy to write open source. However, in a few weeks my caseload should lighten and I definitely should put up or shut up. Thanks for helping me keep it real here.

Tue, 11 Jul 2006 01:01:10, Francis DS, Two apologies...er..comments Expand All | Collapse All

- Sometimes you need to comment why you're doing something one way instead of a more obvious way. There is no way the reason can be found in the code.

- Does Tim consider labels on car dashboards apologies for not designing the dashboard in a more obvious way?

Tue, 11 Jul 2006 19:25:03, Tim Ottinger, Dashboard? Expand All | Collapse All

I never did before. But now that you mention it, i can tell the gas guage from the speedo from the odo without text, which is an important bit of ergonomic engineering.

Tue, 5 Sep 2006 09:50:36, Simon Tamman, Expand All | Collapse All

I write comments to explain use cases as to why a bit of code is there:

e.g. this code is here because of:
Action X, Y Z which leads to this problem. You can now see this code prevents that problem.
I never inline, always in remarks in the method declaration (otherwise the code is less readable).

This means that people consider the use case before deciding to delete the code, rather than just deleting it without sufficient thought in a refactoring sweep.

Wed, 6 Sep 2006 15:14:56, Tim Ottinger, Cases worth considering? Expand All | Collapse All

1) If you cannot make the code more clear, then the comment is acceptable.
2) If you are dealing with freaky undocumented platform features, and can't make the code more clear...
3) To inject humor. Sometimes a little humor is okay. Some comments are funny, and that's not always unprofessional.
4) Somebody in power over you demands that you do comment, and you have no option other than quitting or failing.
5) MFC/COM are such incredible screwups that all code that deals with them should be written as clearly as possible and when it can't be any clearer, then the comment is acceptable.

How is that for meeting halfway?

Tue, 26 Sep 2006 23:36:53, Tim Ottinger, More on my inability to house a single novel idea: *Expand All* | *Collapse All*

This was all covered a long time ago on Ward's wiki.

Front Page | User Guide
root (for global !path's, *etc.*)
SetUp[?]TearDown[?]