

Re: How to understand huge code

Source: <http://coding.derkeiler.com/Archive/General/comp.programming/2008-03/msg00485.html>

- *From:* "jehugaleahsa@xxxxxxxx" <jehugaleahsa@xxxxxxxx>
 - *Date:* Sun, 16 Mar 2008 14:34:01 -0700 (PDT)
-

On Mar 14, 2:52 am, Jack.Thomson...@xxxxxxxx wrote:

Hi All,

I am working with C,C++ since past 1.5 years, I am also pretty good in writing small programs. But when it comes to understanding big projects, (Ex: Say I want to contribute in a open source storage product) I feel very difficult understanding the complete code flow.

Any help on the approach, how to understand huge projects code.

-----htt

I am currently working on an old C++ program that was poorly constructed. It is very difficult to understand code just by following the flow. Indeed, you absolutely must request more information from those who have been working in it. "What is this code doing? Why does it do it that way? What do you think you could do to make it easier for me to understand?" I am constantly asking the original writer of the aforementioned program what his code is doing exactly. Without the original programmer, you can only hope his/her comments are sufficient (they rarely are).

Another big part of understanding code is being able to separate the code that does different things. It is common to find code with intermingling SQL, logic and sub-system calls. Simply gutting out a method and just leaving the logic can lead to amazing clarity. I prefer to gut it and convert the logic to psuedo code. Somehow, writing it in a semi-programming language helps reduce clutter. I tend to replace sub-system calls with an overall summary of the subsystem call, applying the english equivalent to how the parameters are being used and modified. Where the results of a method are used in multiple places, I replace the result variable, again, with the subsystem call summary. For things like SQL, I ask myself what the SQL is retrieving and write something like, "Gets all customer information where their purchases > \$15.00".

I prefer to break down huge code from the top down and then the bottom

Re: How to understand huge code

up when I get stuck. Code on top usually is more simple and has a faster flow. Code on the bottom tends to be an implementation detail, throwing variables around and entering and exiting scopes. In a good system, the names are usually enough to tell you what the method is doing . . . just ignore the rest.

.