Tips for working in a large library? [closed]

Asked 16 years, 3 months ago Modified 12 years, 7 months ago

Viewed 327 times

4

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 5 years ago.

Improve this question

I'm currently working on a quite large library (5M lines of code, in C++ under VS2005, 1 solution and close to 100 projects). Even though we distribute compilation, and use incremental linking, recompilation and relinking after small source modifications takes between a few minutes (usually at least 3) and close to one hour.

This means that our modify code/build/debug cycles tend to be really long (to my taste!), and it's quite easy to lose the 'flow' during a build: there's typically not much time to do anything useful (maybe do a bit of email, otherwise read some article online or a few pages of a book).

When writing new code or doing major refactoring, I try to compile one file at a time only. However, during debugging for example, it really gets on my nerves!

I'm wondering how I could optimize my time? I guess I'm not the only one in that situation: what do/would *you* do?

project-management

Share

Improve this question

Follow

edited May 17, 2012 at 8:15

skaffman

403k ● 96 ● 824 ● 774

asked Sep 1, 2008 at 21:00

OysterD

6,750 ● 5 ● 35 ● 33

3 Answers

Sorted by: Highest score (default)

1

I don't know much about development at that level, but... it seems like it would be a good idea to separate into multiple solutions. You could have a final "pre-ship" step that consolidates them all into a single .dll if you/your customers really insist.

Compare, e.g., to the .NET Framework where we have lots of different assemblies (System, System.Drawing, System.Windows.Forms, System.Xml...). Presumably all of these could be in different solutions, referencing each

other's build results (as opposed to all in a single solution, referencing each other as projects).

Share Improve this answer

Follow

answered Sep 1, 2008 at 21:22

Domenic

113k ● 42 ● 226 ● 273

1

Step by step...

The only solution is to start isolating blocks of code. If you don't have too much implementation leakage (see below **) then start building fachades that isolate the classes behind. Move those clases to a different project and make the fachade load the dlls on startup and redirect the calls to factory methods.

Focus on finding areas/libraries that are fairly stable and split them to isolated library dlls. Building and versioning them separately will help you to avoid integration pains.

I have been on that situation on the past and the only way is to take the task with patience.

By the way, a good side effect of splitting code is that interfaces became cleaner and the output dll size is smaller!!. In our project suffling/reorganizing the code around and reducing the amount of gratuitous includes reduced the final output by 30%.

good luck!

** --> a consumer calling obj->GetMemberZ()->GetMemberYT->GiveMeTheData(param1, param2)

Share Improve this answer

Follow

answered Sep 30, 2008 at 12:35

argatxa

520 ● 4 ● 12

0

@Domenic: indeed, it would be a good thing... However, a whole team's been at it for some time now, and until they succeed we are stuck with a single .dll and something quite monolithic :-(

Share Improve this answer

Follow

answered Sep 1, 2008 at 21:27

OysterD

6,750 ● 5 ● 35 ● 33