Essay on *Mythical Man Month*

I was really taken by the idea of a surgical team. I've often been struck by what a big difference there is in programming efficiency and performance between one programmer and the next, but I've seen few solutions that really deal with that. I've seen some paradigms that remind me of the surgical team, but none that quite went so far. In the past I've done pair programming, writing code while a friend watches over my shoulder making comments, catching bugs, and offering advice. This is very nice, effective, and remeniscant of the surgeon/copilot relationship. Also, in Google, I was exposed to some level of specialization (that is, people performing only one specific *type* of task) and communal testing. This was nice, but as it was implemented at Google it did almost nothing to alleviate the classic problems that slow down large groups.

Using the surgical team approach has almost all the benefits of working alone without most of the drawbacks. The surgeon can work at his own pace and build the project to his own vision, but there's always the copilot or the language lawyer to ask for advice. It's easy to hand off small tasks to the editor, tester, or the toolsmith, without any need for the surgeon to get sidetracked. The copilot and the editor watch for bugs as the code is written, so more bugs are dealt with right away. Finally, if the surgeon leaves, the copilot can take over immediately and a new copilot can train. Of course, employing the surgical team approach means working in a group and not alone. However, there's an administrator to take care of most of the non-programming details, the copilot can attend meetings while the surgeon keeps coding, and the surgeon never has to wait on another programmer if he allocates his team well.

All in all, the surgical team approach seems like a great idea which I would love to try. I'm afraid, though, that it is mostly geared towards medium-sized projects, and wouldn't work as well for very large ones. If a program is too much work for a single programmer, or just very easily divisible, it's clear to see that having more programmers working on the same level might improve efficiency. Then again, if the project is very large and divisible, perhaps each component could have its own surgical team.