Colin Stebbins Gordon (colin)
CS190 – 2/21/07

Mythical Man Month Essay

In chapter 4 of The Mythical Man Month, Brooks discusses the importance of conceptual integrity. I have to say that his treatment of conceptual integrity is lacking. It fails to address the maintenance of software resulting in split designs. He mentions in his example with Reims Cathedral, that eight generations of architects consciously suppressed their own biases in order to produce a building with one single, unifying architecture. His treatment of conceptual integrity's role in software engineering proffers no such example.

Last semester, I took CS167/9, writing all of the major components for a simple operating system. The code totaled over 27000 lines of code, most of which was straightforward support code provided by the course TAs. And several thousand lines of code, maintained by groups of 3 to 7 TAs, each for a period of 3 months, with little communication between generations of TAs, and limited documentation of the internal workings of this support code, is about as coherently structured as one might expect after a number of years.

The code for maintaining linked lists was written as a large collection of C macros. The code for performing equally basic operations on nodes for the VFS layer were all functions. The code base was in C, but much of the support code (and the code we students were expected to write) was structured in a manner which required globally-defined static structs of function pointers for manipulating various structs, essentially passing around VTBLs and instance variables for objects manually (there were rumors that the code base was originally in C++, but I've heard no confirmation of this).

A certain amount of ugliness is necessary when implementing an operating system (when I complained that a trick to make something work in Weenix felt like a cheap hack, he reminded me, "It's an operating system – that's all it is, is one big, giant hack!"), but most of this was sheer whim on the part of whoever was updating a particular section of code for the new year. This made each new assignment more difficult than it needed to be, because not only were we learning a new subsystem of a very complex code base, but each subsystem was written very differently from those we worked on before it, slowing us all down.

There must be some way to maintain the sense of conceptual integrity across years, across groups of people who do not sit down and talk about code base architecture. Documentation seems key. But at least in the portion of The Mythical Man Month which we've read thus far, Brooks has yet to address this.