My response to Dr. Robert’s Extreme Programming Lecture

I have heard of Extreme Programming before, in passing, though I’ve never known much about it. What I read/heard of Extreme Programming did seem fairly beneficial, but until Dr. Robert’s talk, I did not realize how beneficial – and esoteric – Extreme Programming could be.

Several elements of Extreme Programming really stood out for me. I was really amazed by the strength of pair programming: both for avoiding the “how many car accidents are we from disaster?”, avoiding code ownership (and the ego involved), getting on-the-fly code review, and keeping each other on task. I was also struck at how a simple restructuring of the cubicles could suddenly lead to incredible increases in efficiency, friendliness (especially amongst coders and testers), communication (including “residual communication”), and the overall quality of the finished code. The ingenuity of a standup meeting – to keep lengthy speeches down, yet to keep everyone on the same page with regards to current work and obstacles that fellow programmers are facing – also stood out to me.

I have known that refactoring can be a powerful tool even in my own code, but I can definitely see how it is utterly essential to counteract “bit rot” and the accumulating a “big ball of mud” of code over the course of some twenty years of coding and patching. To refactor something so massive, it is absolutely essential to have tests in place: otherwise, how would one dare to touch code so ancient and fragile, without immediately seeing whether a small change has caused the collapse of the whole system? I think that Dr. Robert’s greatest contribution to the company was precisely this: to change the “Culture” of the company to embrace daily
testing BY THE PROGRAMMERS THEMSELVES (not just the designated “debugger”), and to help the company implement hundreds of automatic tests to run each night on the “nightly build”.

I am really impressed by the success that Dr. Robert’s whirling winds of change have brought to the company: both in the sense of producing less buggy code, completing projects by set deadlines, and bringing about harmony in the company. I think that some credit also goes to the company and the programmers themselves: for being willing to give XP a full-hearted try, despite its non-traditional approaches. Overall, I think that Dr. Roberts XP story is a very heartening and encouraging one, and I certainly hope that if I ever work for a company, it too will have the good fortune to realize the necessity for – and the successful implementation of – an XP-like approach.