During Dr. Don Roberts's presentation on Extreme Programming ("XP"), he showed a real life example that he helped to implement. Through his advisor, Dr. Roberts was introduced to the Profitstar Company and its software development team. Dr. Roberts recommended the XP methodology for several reasons. First, the development team had lost the confidence of the company because of bug laden and delayed software. Second, the development team lacked the guidance to succeed. Third, the development team had to deal with legacy code and egos – there was some code that simply belonged to certain programmers. The software development team was successful in implementing the XP methodology. They were able to increase productivity and reduce errors in several ways. First, all the programmers were moved to a central bullpen to foster team work and encourage communication. Second, the software testers were moved near the bullpen and third, other managers assumed the role of the customer when requesting new features. These changes, along with the use of index cards to track requests and pair programming helped restore the software development team. Pair programming allowed the team to have real time code checking and helped to diffuse knowledge throughout the team. Index cards help to define how long tasks would take and provide a visual representation of what needs to be done.

XP is an interesting way to program. From Dr. Roberts’s presentation, I came to understand several aspects of XP that make me support it. First, XP eliminates code ownership because everyone works on everything. This is a great benefit because it helps protect the company from having code that no one is trained to update. Eliminating code ownership also ensures that all programmers have an equal stake in the project and therefore are more likely to be more dedicated to it. I personally think these two reasons would make joining an XP shop easier than a traditional development shop. I also can understand the reasons behind using index cards for feature requests and bug reports. By having the “customer” write down and explain a problem, it becomes much easier to understand and figure out later on. For example, this summer the end users of the system I developed would often report bugs and request features face to face or in passing. This often resulted in me scribbling down what they said and then either not understanding their problem or not remembering what exactly they were trying to do. Using index cards would eliminate this and also provide a paper trail. The one aspect of XP that Dr. Roberts presentation did not convince me of was the pair programming. While his arguments for it make sense, my limited experience with it contradicts what he said. In CS 210 we did pair programming and it did not seem to diffuse knowledge or improve the code. It also seems like a waste of human resources to only have one person actually typing in code. Perhaps if I tried it with a programmer of similar ability it would lead to improved code and mutual understanding of how it works.

I think XP has potential. After working in the Business Systems Department of OTS, I think they could benefit from the XP methodology. First, because of code ownership, there are some systems and processes that no one fully understands. Second, several people are beginning to learn to program in Datatel. Since about half the department has experience in Datatel programming, pair programming seems like it would be a logical method to disperse knowledge and bring everyone to the same level of Datatel programming experience. Third, I think a bullpen would help almost any development team because it forces face to face interaction. This both improves communication and forces the team to spend time as a group.

Overall, this presentation showed me that XP can work. While I’m still not completely sure how well it really works, it appears to work. I think it would be interesting to spend a week in the bullpen to see what the environment is really like in an XP shop.