

## eXtreme Programming: Net Objective's View

Essentially, we are in total agreement with the principles of eXtreme Programming, including:

- A strong team is important
- People should accept responsibility, not be given it
- People should do what they are good at
- You cannot control change
- You cannot control all aspects of development
- Testing is good
- Design is good
- Simplicity is good
- Quick cycles are needed to get feedback
- Frequent integration is good

We also completely agree with many of the strategies that are espoused in XP to implement these principles, including:

- Up front testing
- The planning game
- Programmers being responsible for programming things
- Customers being responsible for selecting and specifying features
- Keeping things simple (but we may disagree on what simple is)
- Refactoring

While we agree with these, we find that we have alternatives to some of the XP practices, including:

- Paired programming
- What we consider inadequate documentation
- Design as you go

There are other issues we have that we will write up later. These involve:

- Using design patterns in XP
- Using frameworks

Our primary concern with XP is that it only works for very small projects (10 or less). Even then, there are some problems with it if your team is not really good. I know many teams that have had great success with XP, but without exception (at least the ones I've known), they either have a *very* strong leader (e.g., Kent Beck, Ron Jeffries, Bob Martin, Martin Fowler) or fairly strong teams. To improve the quality of teams, we integrate design pattern training into our XP training.

To summarize, in our view, the strongest features of XP are:

- Avoiding paralysis by analysis
- Getting quick feedback
- Business people deciding what functionality is needed
- Never implementing any functionality on anticipation of its use
- The Planning Game
- Having developers decide on costs
- Testing up-front
- Writing re-factorable code