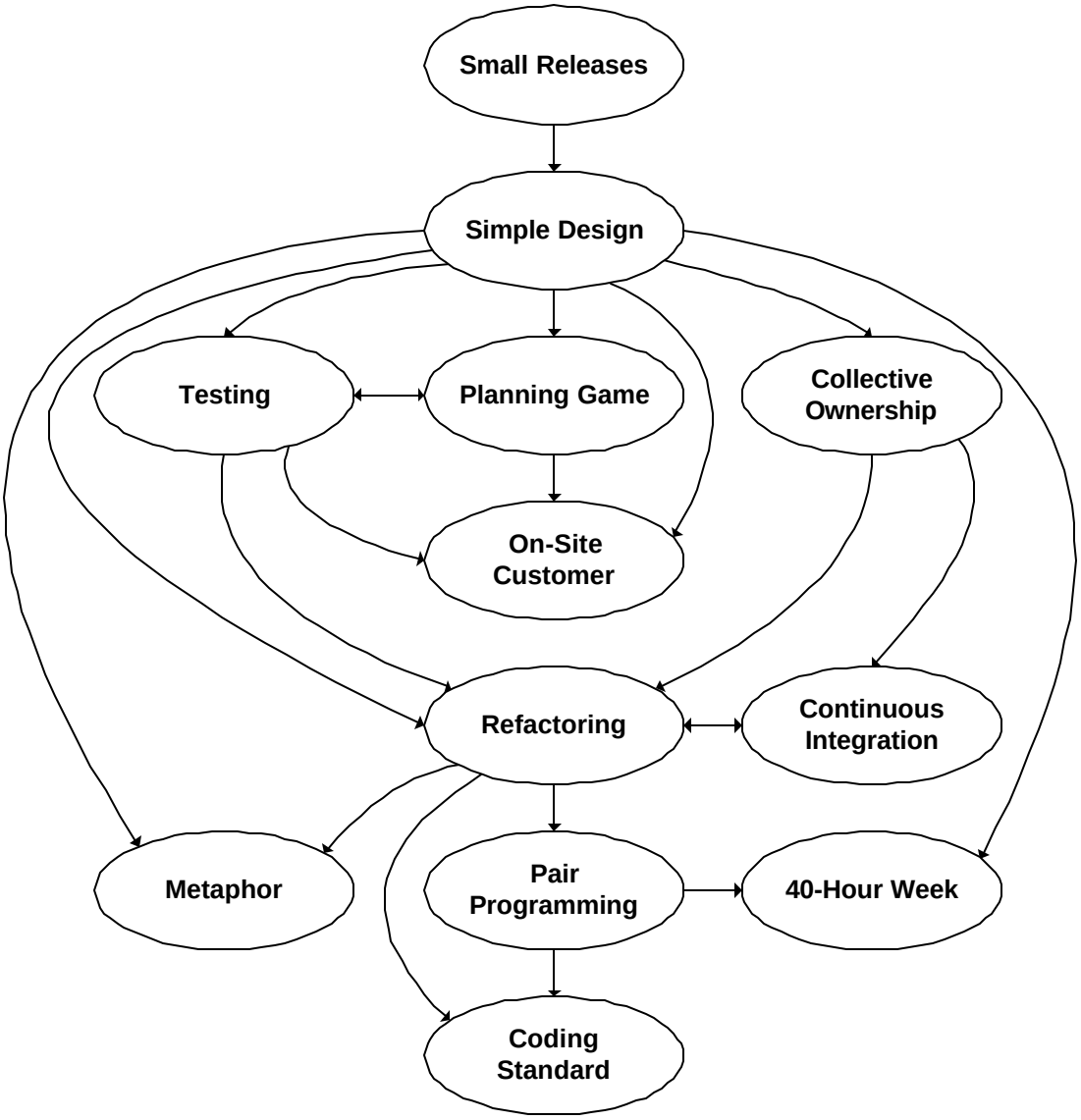
The Extreme Programming Playbook

Every team on every playing field of software development moves and competes within the boundaries of a written or unwritten playbook. Below are the extreme plays in the playbooks of extreme software teams.



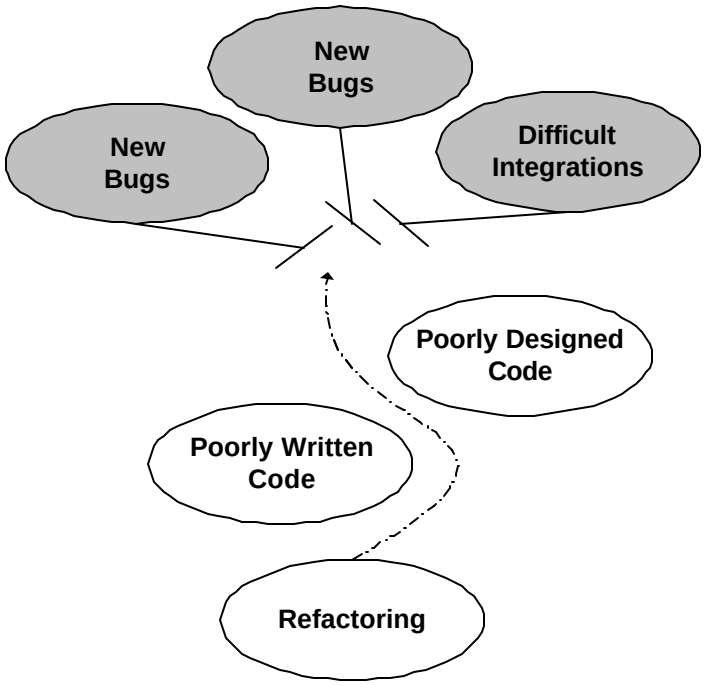
The first play, upon which all the others depend, is Small Releases. The software team makes continuous forward motion by producing an embryonic version of a system, releasing it for immediate customer feedback, adding crucial functionality with every new release and continuously learning from feedback.

To maximize forward motion, every play has a Simple Design, allowing a team to do the most with the least amount of effort. Simplicity permeates every action: the design of the user interface, the programming of the business algorithms, the smallest sections of code.

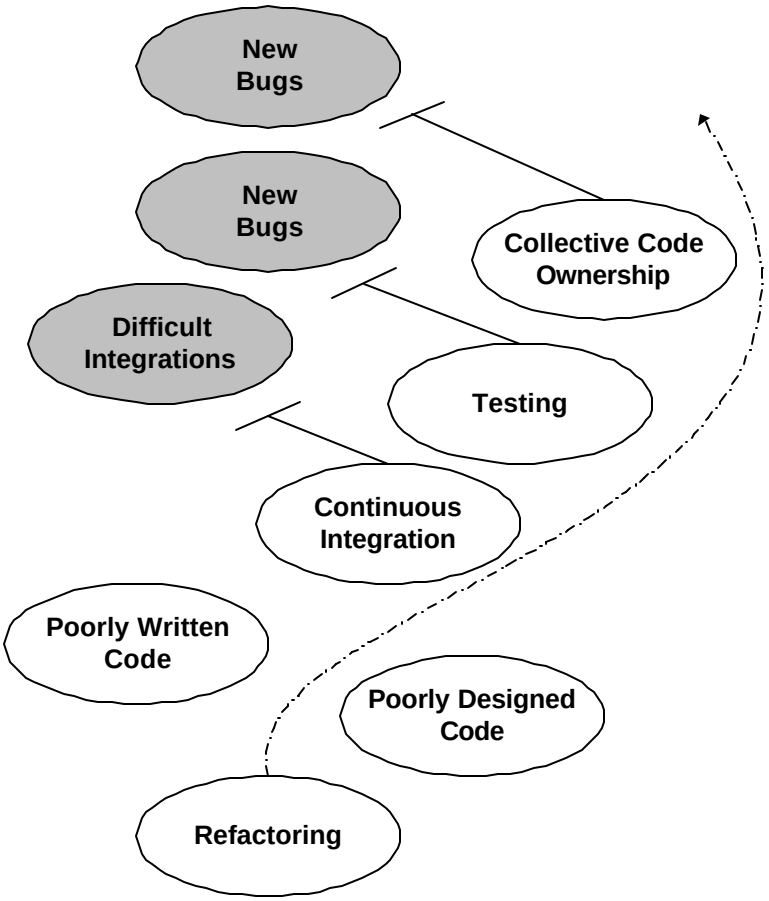
A team successfully executes Small Releases and Simple Design with seven coordinated plays: The Planning Game, Testing, On-Site Customer, Collective Code Ownership, Refactoring, Metaphor and 40-Hour Week.

The Planning Game provides short, focused game plans, which the team executes at regular intervals. A relentless focus on Testing helps the team satisfy each of the game plan's stated goals. The On-Site Customer helps The Planning Game and Testing go according to plan. Collective Code Ownership places team play over individual play for superior performance. Refactoring is the merciless drive towards simplicity, clear communication and the removal of obstacles. Metaphor, one of the hardest plays in the book, provides insight into and new development of patterns of play. The 40-Hour Week keeps players strong and on balance during the game.

Some of these extreme plays cannot be executed effectively without a coordinated coupling between defensive and offensive plays. Consider the following play:



A team can't successfully charge ahead with Refactoring without the defensive support of Testing and Continuous Integration or the offensive freedom provided by Collective Code Ownership:



Merciless refactoring has the potential to maximize a team's efficiency but it can also cause unwanted clashes between fellow team players. To minimize clashes, players Continuously Integrate their maneuvers with the team's latest position. Yet even this defensive play can fail if player performance is poor: a rookie can make serious blunders, a veteran can have a bad day. Extreme teams don't compensate for poor solo play, they elevate the game of *all* players by coupling them together. A veteran pairs with a rookie, a seasoned veteran teams up with an up-and-coming player, two rookies pair together and all players rotate pairs continuously. In this way, mastery of the game spreads, as pairs, not solo players, share the glory of testing, designing, analyzing, programming, refactoring and continuously integrating their contributions. To ensure that each pair is fully engaged on the playing field, everyone follows the 40-Hour Week rule.

Both Pair Programming and Refactoring help make a Coding Standard possible. This standard brings a consistent style of play to the game.

Refactoring also helps a team move towards an ever-elusive Metaphor. While teams rarely know their metaphor early in a game, they can discover one over time, sometimes with the help of Refactoring.

There are many ways to explain the network of plays in the Extreme Programming Playbook. This has been an introduction to those plays. For more information, please see Kent Beck's book, *Extreme Programming Explained*.